

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Сервіс інтерактивна мапа для пошуку та пропозиції роботи»

Виконав студент групи ІПЗ-22-4
ЗАЙЦЕВ Олександр Ігорович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЗАЙЦЕВ Олександр Ігорович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Сервіс інтерактивна мапа для пошуку та пропозиції роботи»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Вихідні дані для роботи: процес автоматизації взаємодії між роботодавцями та
шукачами зайнятості, який використовує інтерактивну мапу для відображення
робіт та підвищення ефективності працевлаштування.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	01.03.2026р.	01.03.2026р
2	Морозов А.В.	01.03.2026р	01.03.2026р
3	Морозов А.В.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з додатку сервісу пошуку та пропозиції роботи на основі інтерактивної мапи та пояснювальної записки. Пояснювальна записка до випускної роботи містить 99 сторінок, 68 ілюстрацій та 15 таблиць.

Метою роботи є розробка геоінформаційного сервісу для автоматизації взаємодії між роботодавцями та шукачами роботи з використанням інтерактивних мап для візуалізації робіт у реальному часі.

У першому розділі проведено комплексне дослідження ринку тимчасової зайнятості та організації фрілансу. Виконано критичний огляд та порівняльний аналіз наявних програмних агрегаторів і платформ оголошень. На основі виявлених логістичних та фінансових обмежень аналогів сформульовано задачу дослідження, а також аналітично обґрунтовано вибір модульної монолітної архітектури та обраного високопродуктивного технологічного стека.

У другому розділі деталізовано функціональні та нефункціональні вимоги сервісу. Розроблено модульну схему реляційної бази даних та змодельовано основні алгоритми системи.

У третьому розділі викладено особливості програмної реалізації. Описано порядок розгортання та оркестрацію інфраструктури. Наведено детальні приклади графічних інтерфейсів та результати комплексного тестування працездатності, надійності й пропускну здатності сервера.

В результаті виконання роботи розроблено геоінформаційну вебплатформу, що забезпечує повний життєвий цикл безпечної координації та автоматизації взаємодії між замовниками й виконавцями одноразових робіт.

Ключові слова: ВЕБСЕРВІС, ІНТЕРАКТИВНА МАПА, ГЕОЛОКАЦІЯ, .NET, REACT, TURBOREPO, ВЕКТОРНІ ТАЙЛИ, ПОШУК РОБОТИ.

					ІПЗ.КР.Б – 121 – 26 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Зайцев О.І.			Сервіс інтерактивна мапа для пошуку та пропозиції роботи			Літ.	Арк.	Аркушів	
Перевір.		Морозов А.В.								3	99
Керівник								Житомирська політехніка, група ІПЗ-22-4			
Н. контр.											
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The bachelor's thesis consists of a job search and offer service application based on an interactive map and an explanatory note. The explanatory note for the graduation thesis contains 99 pages, 68 illustrations, and 15 tables.

The aim of the thesis is to develop a geographic information service to automate the interaction between employers and job seekers using interactive maps for real-time job visualization.

The first chapter provides a comprehensive study of the temporary employment market and freelance organization. A critical review and comparative analysis of existing software aggregators and classified advertisement platforms were performed. Based on the identified logistical and financial limitations of analogs, the research objective was formulated, and the choice of a modular monolithic architecture alongside a high-performance technology stack was analytically justified.

The second chapter details the functional and non-functional requirements of the service. A modular schema of the relational database was developed, and the core algorithms of the system were modeled.

The third chapter outlines the specifics of the software implementation. It describes the deployment process and infrastructure orchestration. Detailed examples of graphical user interfaces and the results of comprehensive testing for server functionality, reliability, and throughput are provided.

As a result of the research, a geographic information web platform was developed, ensuring the full lifecycle of secure coordination and automated interaction between clients and contractors of one-time jobs.

Keywords: WEB SERVICE, INTERACTIVE MAP, GEOLOCATION, .NET, REACT, TURBOREPO, VECTOR TILES, JOB SEARCH.

					ІІЗ.КР.Б – 121 – 26 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Зайцев О.І.			Сервіс інтерактивна мапа для пошуку та пропозиції роботи	Літ.		Арк.	Аркушів
Перевір.		Морозов А.В.						3	99
Керівник						Житомирська політехніка, група ІІЗ-22-4			
Н. контр.									
Зав. каф.		Вакалюк Т.А.							

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ	9
1.1. Загальна характеристика проблеми	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури системи	15
1.4. Обґрунтування вибору інструментальних засобів	18
Висновки до першого розділу	22
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ІНТЕРАКТИВНОЇ МАПИ ДЛЯ ПОШУКУ ТА ПРОПОЗИЦІЇ РОБОТИ	24
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	24
2.2. Розробка бази даних системи	28
2.3. Проєктування та реалізація алгоритмів роботи системи	46
2.4. Реалізація функціоналу	50
Висновки до другого розділу	55
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ ІНТЕРАКТИВНОЇ МАПИ ДЛЯ ПОШУКУ ТА ПРОПОЗИЦІЇ РОБОТИ	57
3.1. Порядок розгортання та налаштування параметрів платформи	57
3.2. Структура інтерфейсу мобільного додатку	60
3.3. Структура інтерфейсу панелі адміністратора	82
3.4. Тестування вебплатформи	85
Висновки до третього розділу	89
ВИСНОВКИ	90
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	91
ДОДАТКИ	95

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - прикладний програмний інтерфейс

БД - база даних

ПЗ - програмне забезпечення

СУБД - система управління базами даних

DTO - об'єкт передачі даних, що використовується для інкапсуляції даних та їх відправлення між підсистемами

HTTP / HTTPS - протокол передавання гіпертексту

JSON - текстовий формат обміну даними

DI - впровадження залежностей; архітектурний шаблон проектування, що використовується для забезпечення слабкої зв'язності між компонентами системи (Dependency Injection)

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми полягає у зростаючому попиті на сервіси для швидкого пошуку виконавців одноразових фізичних робіт. У сучасних умовах стандартні платформи з довготривалими вакансіями не задовольняють потребу в оперативному залученні працівників для вирішення побутових завдань на місці. Сервіс допоможе робітникам знаходити нові замовлення та надасть швидкий доступ до виконавців роботи клієнтам. Створення спеціалізованої платформи на основі геолокації суттєво покращить ефективність взаємодії двох сторін, а також дозволить повністю автоматизувати координацію між ними, забезпечуючи прозорість фінансових розрахунків і безпеку кожної угоди.

Стрімкий розвиток мобільних технологій та широке охоплення цифровими послугами створюють сприятливі умови для інтеграції таких рішень у повсякденне життя користувачів. Крім того, сьогодні спостерігається стрімке збільшення кількості людей, для яких формат часткової зайнятості є найбільш привабливим способом заробітку або отримання додаткового доходу. Паралельно з цим з боку споживачів постійно зростає попит на замовлення різноманітних послуг. Таким чином, розробка та впровадження подібного сервісу безпосередньо відповідає сучасним запитам суспільства, допомагаючи ефективно збалансувати масовий інтерес до гнучкого графіку з боку виконавців та зростаючі потреби замовників.

Метою випускної кваліфікаційної роботи бакалавра є розробка програмного комплексу для пошуку та управління короткостроковими фізичними роботами на основі геолокації із використанням сучасних архітектурних підходів.

Завданням проекту став аналіз існуючих аналогів сайтів для пошуку тимчасової або повної зайнятості, та виявити унікальний функціонал, який пропонуватиме сервіс. Спроектувати архітектуру системи та базу даних, за допомогою методів моделювання. Реалізувати програмне забезпечення впровадивши можливість пошуку робіт та робітників за допомогою

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

інтерактивної мапи. Розгорнути сервіс у хмарному середовищі та протестувати систему.

Об’єктом дослідження є процеси проектування та розробки веб-орієнтованих систем для організації ринку тимчасової та одноразової зайнятості.

Предметом дослідження є функціональні можливості, архітектурні рішення та алгоритми роботи сервісу інтерактивної мапи, що забезпечують ефективну взаємодію між замовниками та шукачами одноразової роботи.

Практичне значення одержаних результатів полягає у створенні працездатної вебплатформи для оперативного пошуку виконавців одноразових фізичних робіт на основі геолокаційних даних. Розроблений сервіс може бути безпосередньо використаний замовниками для швидкого вирішення побутових чи термінових завдань, а також робітниками - для ефективного пошуку підробітку в режимі реального часу.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ

1.1. Загальна характеристика проблеми

У межах кваліфікаційної роботи необхідно дослідити предметну область організації пошуку та пропозиції короткочасної роботи на основі геолокаційних даних та сформуванню уявлення про задачу для подальшого виконання. Постановка задачі охоплює визначення цілей і обмежень системи, аналіз потреб замовників і виконавців, а також проектування базового функціоналу: інтерактивної мапи, алгоритмів підбору оголошень та безпечної оплати.

Користувачами платформи є замовники, які шукають оперативне вирішення термінових завдань, і виконавці, зацікавлені у швидкому підробітку через виконання робіт. Архітектура системи має забезпечити чітке розділення ролей. Кожен обліковий запис наділяється одним типом прав замовника або виконавця, що визначає відповідний сценарій взаємодії з платформою.

Основною проблемою взаємодії замовників та виконавців є високий ризик шахрайства, зумовлений специфікою укладання угод. Зазвичай координація та фіксація умов відбуваються через неформальне обговорення у розрізних сервісах (месенджерах, телефонних дзвінках), а фінансові розрахунки здійснюються за фактом уже виконаної роботи. За такої схеми сторони змушені покладатися лише на взаємну довіру, оскільки відсутні інструменти перевірки надійності партнерів. Неорганізованість цих процесів поза єдиною інтерактивною платформою унеможливорює моніторинг завдань за локацією, веде до втрати історії домовленостей та ускладнює вирішення суперечок.

Метою кваліфікаційної роботи є створення централізованої вебплатформи для швидкого пошуку виконавців або замовників на основі їхнього розташування. Система автоматизує весь життєвий цикл угоди - від пошуку на інтерактивній мапі до фінальних розрахунків і покрокового

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

контролю виконання. Для зручності взаємодії користувачів передбачено внутрішній месенджер, фінансову безпеку гарантує логіка контрактів, реалізована через платіжні інструменти Monobank. Платформа не замінює собою спеціалізовані геоінформаційні системи чи банківські інституції, а також не відстежує фізичний процес виконання робіт поза межами програми. Таке чітке обмеження функціонала дозволяє зосередитися на головній цінності продукту - створенні безпечного, прозорого та мобільного середовища для короткочасних робіт.

На основі аналізу виявлених проблем сформульовано перелік функціональних вимог до платформи. Вона має забезпечувати функціонал публікації та керування оголошеннями про термінові фізичні роботи із зазначенням точного місця на карті, опису завдання, термінів виконання та пропонованої оплати. Система повинна підтримувати алгоритми динамічного пошуку та фільтрації замовлень у реальному часі залежно від встановленого радіуса віддаленості від користувача. Важливою вимогою є реалізації механізму безпечної оплати та замороження коштів, коли гроші замовника затримуються на рахунку системи й перераховуються виконавцю лише після підтвердження успішного завершення роботи. Крім того, платформа має включати систему внутрішніх чатів для швидкого узгодження деталей завдання, покроковий супровід угоди, а також механізм взаємного оцінювання та збору відгуків для формування рейтингу надійності користувачів.

Крім функціональних вимог, платформа має нефункціональні вимоги, серед яких розробка у вигляді мобільного додатка для забезпечення максимальної зручності. Інтерфейс додатка має бути повністю адаптованим під різні розширення екранів смартфонів та планшетів, підтримувати нативні жести та забезпечувати високу швидкість завантаження інтерактивної мапи й обробки пошукових геозапитів навіть в умовах нестабільного мобільного інтернету. Система має гарантувати безпечну взаємодію із зовнішніми картографічними та платіжними сервісами через захищені канали зв'язку,

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

надійне збереження персональних даних на пристрої та сервері, а також стійкість до високих навантажень за рахунок асинхронної обробки подій.

Таким чином, основними функціями платформи є геолокаційне відображення та пошук замовлень та робітників на інтерактивній мапі, координатія замовників і виконавців, ведення перемовин у внутрішньому чаті, оформлення умов короткострокових угод із безпечною оплатою, а також супровід процесу виконання робіт від початку до фінального підтвердження із фіксацією дій кожної зі сторін. Визначені функції та потреби користувачів становлять основу для формування архітектури системи та її подальшого проєктування і реалізації.

1.2. Аналіз аналогів програмного продукту

Розробка сервісу інтерактивна мапа для пошуку та пропозиції роботи потребує глибокого аналізу наявних рішень на ринку. Оскільки ідея локального фрілансу для майстрів активно розвивається, проведемо аналіз популярних платформ, що працюють за традиційною моделлю.

Kabanchik.ua - провідна українська платформа для пошуку виконавців на різноманітні завдання (рис. 1.1).

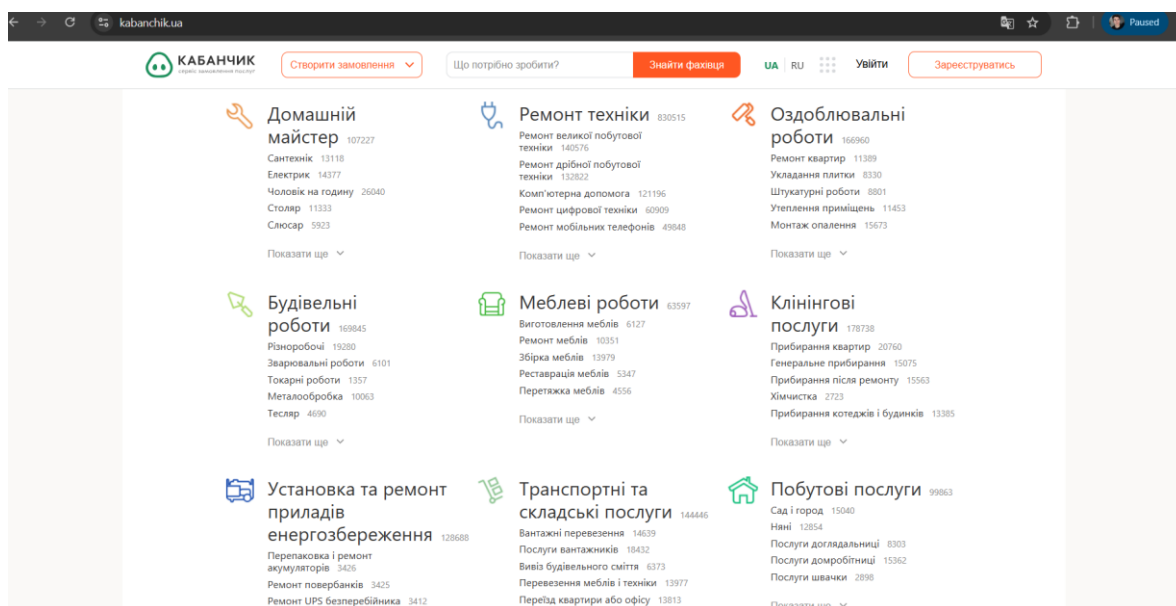


Рис. 1.1. kabanchik.ua

Функціонал платформи забезпечує розгалужену структуру категорій послуг, яка охоплює як дрібний побутовий ремонт (наприклад, послуги

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

сантехніків чи електриків), так і складніші цифрові чи клінінгові завдання. Взаємодія з користувачами оптимізована завдяки наявності мобільних додатків для операційних систем iOS та Android, що дозволяє оперативно отримувати сповіщення про нові замовлення. Безпека та прозорість угод підтримуються впровадженою системою двосторонніх рейтингів та текстових відгуків, які формують рівень довіри до виконавця на основі його попереднього досвіду.

Водночас архітектура сервісу передбачає значне фінансове навантаження на виконавців у вигляді високої комісії з кожного успішно виконаного замовлення, що знижує маржинальність роботи для самозайнятих майстрів. Окрім цього, процес реєстрації та допуску до виконання завдань ускладнений тривалою процедурою верифікації нових профілів. З точки зору інтерфейсних рішень, Kabanchik.ua функціонує переважно як текстовий агрегатор оголошень, орієнтований на класичні списки. Такий підхід суттєво обмежує можливості швидкого візуального пошуку замовлень «поруч із собою» і не дозволяє ефективно оцінити логістичну доцільність виконання роботи без переходу на сторінку конкретного завдання.

OLX (Робота) - один із найбільших сервісів оголошень в Україні (рис. 1.2).

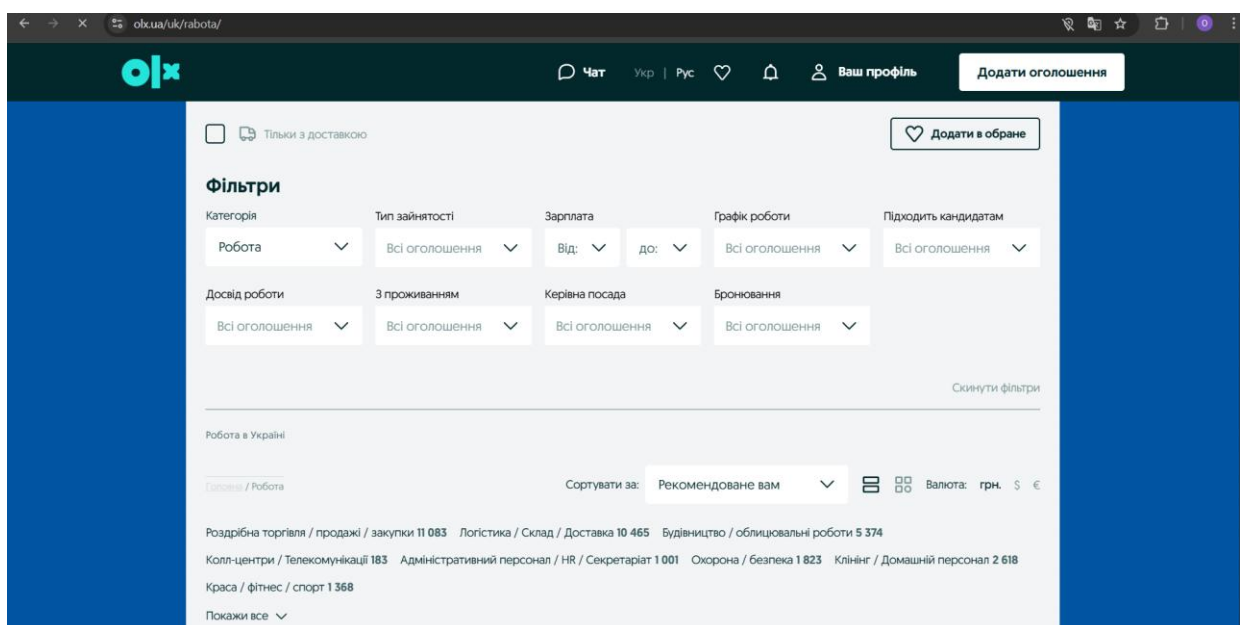


Рис. 1.2. olx.ua

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Основна функціональна перевага цього рішення полягає у величезній базі активних користувачів та максимальній спрощеності процесу публікації оголошень. Інтерфейс подачі пропозицій є інтуїтивно зрозумілим, що мінімізує поріг входження для майстрів будь-якого рівня кваліфікації. Проте, будучи універсальною дошкою оголошень, OLX не має функціональної спеціалізації на терміновій, одноразовій або погодинній праці. Головним недоліком платформи у розрізі локального фрілансу є повна відсутність інструментів динамічного відображення геолокації вакансій та виконавців на інтерактивній мапі. Користувачі змушені обмежуватися текстовими фільтрами за містами або районами. Через таку універсальність сервіс не здатний забезпечити необхідну оперативність для вузькоспеціалізованих майстрів, для яких критично важливим фактором є миттєве візуальне визначення відстані до клієнта з метою мінімізації часу та витрат на дорогу.

Upwork - глобальна фріланс-біржа, що задає стандарти віддаленої взаємодії (рис. 1.3).

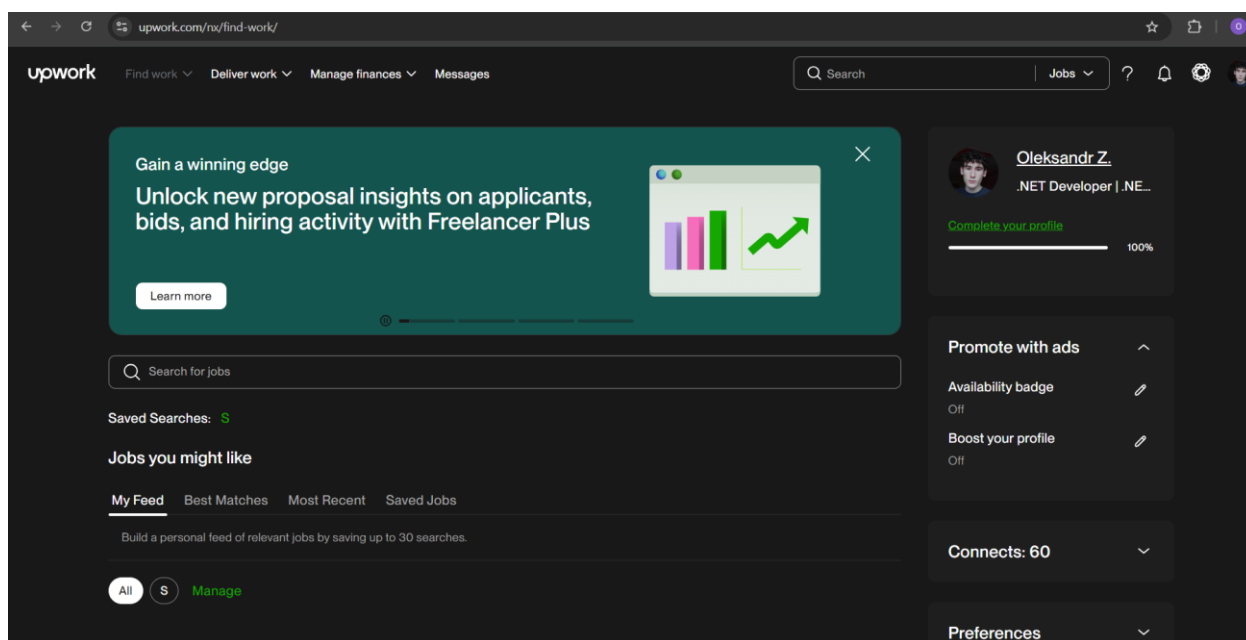


Рис. 1.3. upwork.com

Функціональне наповнення платформи включає передові професійні інструменти для комплексного управління життєвим циклом замовлень: від трекінгу робочого часу до автоматизованого формування транзакцій. Безпека

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

фінансових операцій забезпечується багаторівневою системою ескроу-рахунків та абсолютно прозорими тарифними планами.

Незважаючи на технологічну досконалість, бізнес-модель та функціонал Upwork повністю орієнтовані на дистанційну зайнятість та інтелектуальну працю. Платформа позбавлена будь-яких механізмів локальної прив'язки для виконання завдань, що потребують фізичної присутності. Складний інтерфейс, орієнтований на довгострокові контракти та детальне технічне проектування, є надлишковим і незручним для швидкої координації побутових чи екстрених локальних послуг у реальному часі.

Виходячи з проведеного аналізу функціональних можливостей, архітектурних обмежень та практичного досвіду експлуатації розглянутих платформ, сформуємо порівняльну таблицю програмних аналогів (табл. 1.1) для наочного визначення вимог до розроблюваного сервісу.

Таблиця 1.1

Порівняння платформ для пошуку виконавців

Характеристика	Kabanchik	OLX Робота	Upwork	Розроблюваний сервіс
Наявність інтерактивної мапи	-	-	-	+
Орієнтація на фізичну працю	+	+	-	+
Спеціалізація на локальному фрілансі	+	-	-	+
Радіусний пошук роботи	-	-	-	+

Переглянувши ринок, можна стверджувати, що конкуренти здебільшого пропонують статичні списки або фільтрацію за містом, що не дозволяє шукачу роботи точно оцінити відстань до об'єкта. Впровадження радіусного пошуку

дозволяє користувачу вказати конкретну відстань (наприклад, у межах 2-5 км), що важливо для зручного пошуку роботи.

1.3. Вибір архітектури системи

Одним із ключових етапів проєктування програмного комплексу є вибір архітектури системи. Архітектурні рішення визначають структуру програмного забезпечення, принципи взаємодії між клієнтом та серверною частиною, механізми обробки даних, а також можливості масштабування, відмовостійкості та подальшого розвитку платформи в умовах високої інтенсивності запитів.

З урахуванням специфіки предметної області, було розглянуто основні підходи до побудови архітектури програмного комплексу: монолітну архітектуру, мікросервісну архітектуру, та модульну монолітну архітектуру.

Класичний моноліт передбачає збірку та розгортання всього функціоналу додатка у вигляді єдиної логічної одиниці, яка виконується всередині одного системного процесу. У такій структурі всі компоненти, включаючи автентифікацію, бізнес-логіку та роботу з даними, мають безпосередній доступ один до одного через прямі виклики в оперативній пам'яті [29].

Інформаційна модель традиційного моноліту базується на використанні спільної централізованої бази даних, де будь-яка частина коду може виконувати запити до довільних таблиць. Головною рисою цього підходу є відсутність жорстких технологічних бар'єрів між функціональними доменами, що на початкових етапах забезпечує найвищу швидкість розробки та простоту локального тестування. Кодова база залишається цілісною, а для запуску системи достатньо налаштувати одне серверне оточення. Через відсутність внутрішніх бар'єрів кодова база з часом перетворюється на заплутані взаємозв'язки, де мінімальна зміна в одному місці непередбачувано руйнує роботу інших частин системи. Схематичне зображення монолітної архітектури наведено на рис. 1.4.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

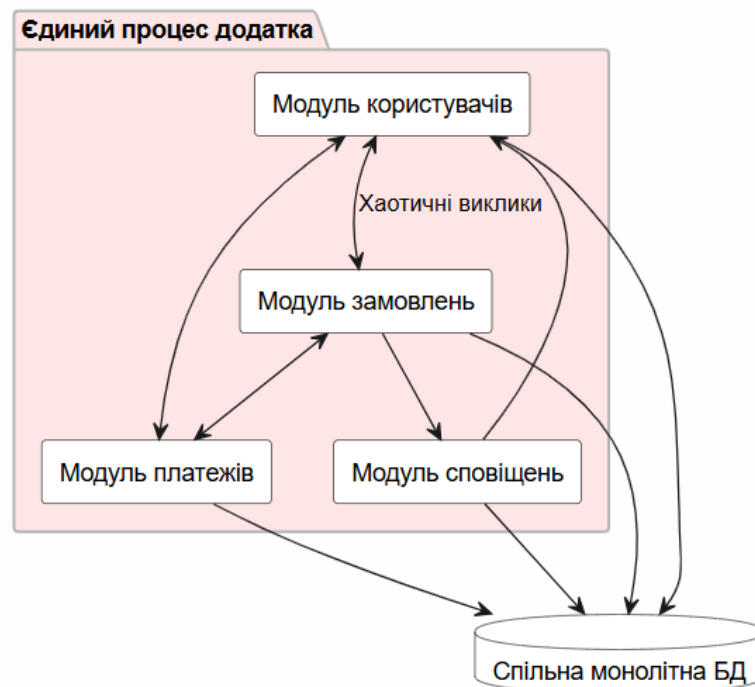


Рис. 1.4. Традиційний моноліт.

Мікросервісний підхід повністю відмовляється від ідеї єдиного процесу на користь розподіленої системи, де кожен бізнес-домен реалізується як повністю незалежний додаток. Кожен мікросервіс створюється, тестується, розгортається та масштабується автономно, часто з використанням індивідуального технологічного стеку, що найкраще підходить для конкретного завдання [1].

Зв'язок між сервісами стає розподіленим і здійснюється виключно через мережеві протоколи, такі як синхронні HTTP/gRPC запити або асинхронні брокери повідомлень [30]. Організація даних підпорядковується концепції власної бази даних для кожного мікросервісу. Спільне використання сховищ виключається, що вимагає впровадження специфічних патернів для підтримки консистентності даних у розподіленому середовищі. Така архітектура забезпечує максимальну відмовостійкість та дозволяє паралельно працювати над продуктом великій кількості незалежних команд.

Проте, цей підхід створює інфраструктурне ускладнення, яке змушує розробників боротися з мережевими затримками, нестабільністю зв'язку та

складними механізмами підтримки узгодженості даних у різних базах. Схематичне зображення мікросервісної архітектури наведено на рис. 1.5.

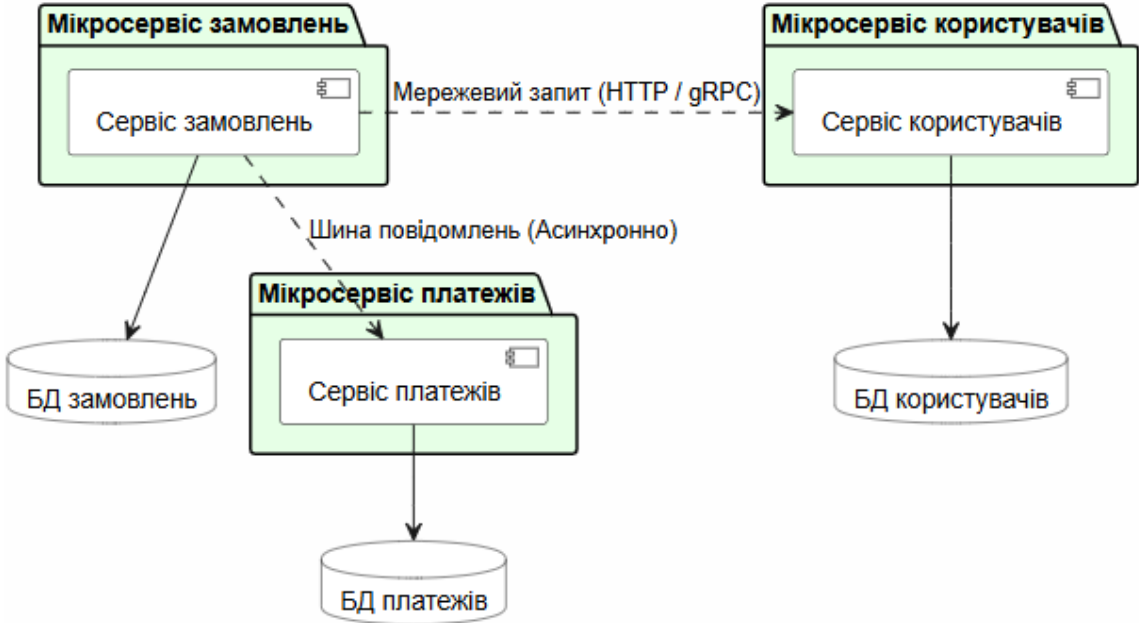


Рис. 1.5. Мікросервісна архітектура.

Модульний моноліт так само розгортається і виконується як єдиний програмний процес, що спрощує інфраструктурне супроводження та релізні цикли. Проте внутрішня організація коду підпорядкована принципу суворої інкапсуляції. Весь додаток розділений на автономні модулі, кожен з яких відповідає за окрему бізнес ціль. Кожен модуль має чітко визначений публічний контракт, який слугує єдиною точкою входу для інших частин системи [2].

Взаємодія між модулями відбувається виключно через ці інтерфейси у межах одного процесу, без виходу в мережу. На рівні збереження даних діє правило ізоляції: кожен модуль володіє власною схемою у спільній базі даних або окремим набором таблиць, а прямі міжмодульні запити до чужих даних суворо заборонені програмною архітектурою. Це створює надійний баланс між високою швидкістю інпроцесних обчислень та логічною чистотою системи (рис. 1.6).

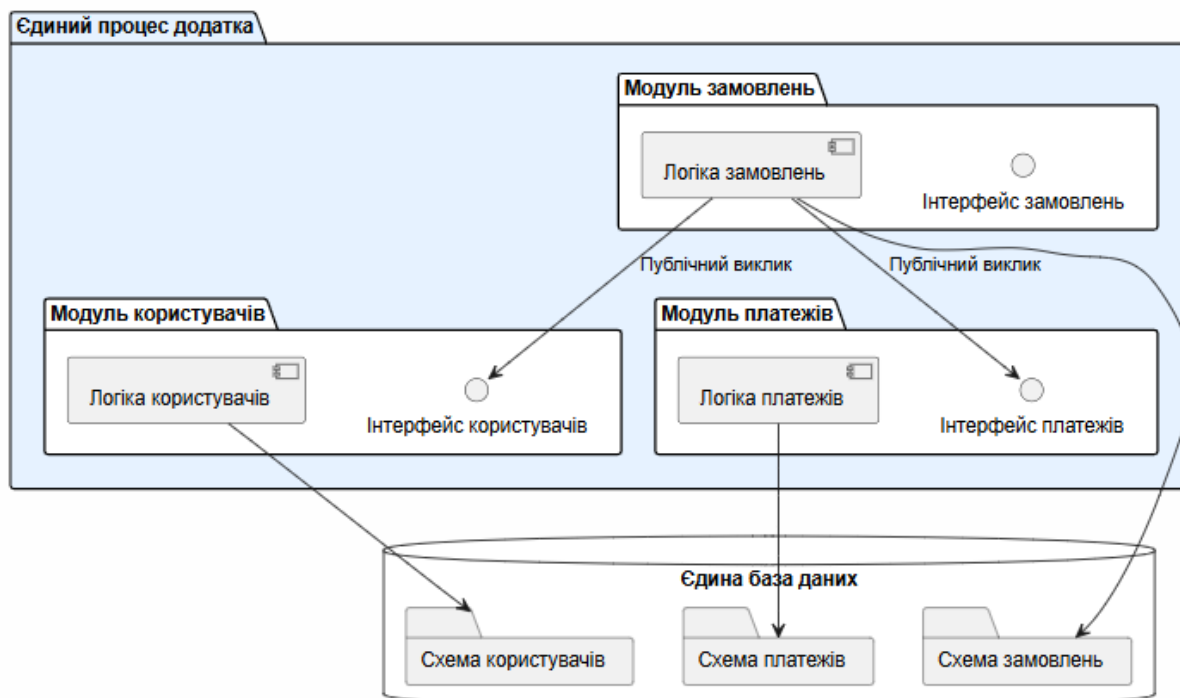


Рис. 1.6. Модульна монолітна архітектура.

Розглянувши та всебічно проаналізувавши особливості побудови кожної з наведених архітектур, для реалізації даного проекту було обрано саме модульний моноліт. Цей варіант визнано найбільш оптимальним, оскільки він дозволяє уникнути надмірної інфраструктурної складності та мережевих ризиків мікросервісів, але водночас надійно захищає систему від втрати контролю над кодом, які притаманні традиційному моноліту. Таким чином, обрана архітектура забезпечує високу швидкість розробки та простоту розгортання на початкових етапах, зберігаючи чітку ізоляцію контекстів для потенційного масштабування системи у майбутньому.

1.4. Обґрунтування вибору інструментальних засобів

Для реалізації розроблюваного програмного комплексу було обрано сучасний та високоефективний технологічний стек, який повністю задовольняє вимогам продуктивності, масштабованості та оперативності обробки даних.

Основою серверної частини платформи обрано технологію .NET, зокрема її актуальну версію .NET 10. Вона забезпечує одні з найвищих показників швидкодії та надійності серед сучасних рішень для розробки

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

серверних систем. Платформа демонструє відмінну продуктивність завдяки оптимізованому виконанню коду, ефективному управлінню пам'яттю та високій швидкості обробки запитів, що гарантує стабільну роботу та швидку відповідь сервера навіть за умов інтенсивного навантаження.

Сучасна екосистема .NET проектувалася як повністю кросплатформена інфраструктура з потужним пакетним менеджером NuGet. Це дозволяє безперешкодно запускати, масштабувати та підтримувати сервіси у середовищах Linux, macOS та Windows, а також у Docker-контейнерах без втрати продуктивності. Централізований репозиторій NuGet спрощує керування залежностями, забезпечуючи швидку інтеграцію перевірених бібліотек, сувору автентифікацію пакунків та надійний захист від вразливостей в інфраструктурі проекту.

Для побудови API застосовується фреймворк ASP.NET Core, який містить вбудовані інструменти для автентифікації, авторизації та захисту даних. Використання технології ASP.NET Core Minimal APIs дозволяє створити легковагий, високопродуктивний API з мінімальними накладними витратами на маршрутизацію запитів, що є критично важливим для швидкої обробки координаційних та пошукових запитів користувачів [3]. Платформа має високий потенціал для розширення завдяки гнучким механізмам інверсії управління (Dependency Injection) та модульній структурі обробки запитів за допомогою middleware, що дозволяє підтримувати чистоту коду та архітектурну гнучкість при зростанні системи.

Для організації процесу розробки, конфігурування та локального тестування розподіленої системи застосовано сучасний оркестраційний фреймворк .NET Aspire. Ця платформа виступає в ролі інтеграційного середовища, яке кардинально спрощує керування життєвим циклом та взаємодією між модулями платформи, базами даних та об'єктним сховищем [4].

Оркестрація додатків у .NET Aspire дозволяє описувати всі компоненти інфраструктури в єдиному проекті AppHost за допомогою коду C#. Це усуває

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

потребу вручну запускати окремі контейнери чи налаштовувати складні скрипти розгортання для локального середовища. Фреймворк автоматично керує запуском сервісів, реляційної бази даних PostgreSQL та хмарних емуляторів для зберігання файлів, забезпечуючи їхню синхронізацію та правильну послідовність активації.

Окремою важливою перевагою платформи є вбудована панель моніторингу (Aspire Dashboard), яка функціонує за стандартами OpenTelemetry. Вона надає розробнику наочний інтерфейс для відстеження стану всієї розподіленої системи в режимі реального часу, включаючи збір інформації про продуктивність, централізоване логування та трасування запитів. Це дозволяє миттєво знаходити помилки та аналізувати затримки під час взаємодії між сервісами, що суттєво підвищує швидкість розробки, тестування та налагодження системи.

Клієнтську частину програмного комплексу, відповідно до нефункціональних вимог щодо створення мобільного додатка, вирішено реалізувати за допомогою фреймворку React Native із використанням платформи Expo. Такий вибір обґрунтований можливістю розробки кросплатформеного мобільного додатка для операційних систем Android та iOS на основі єдиної бази коду, що значно скорочує час розробки. Платформа Expo надає нативний доступ до апаратних засобів мобільних пристроїв, зокрема до модулів геолокації, а також забезпечує високу швидкість рендерингу інтерактивних карт, що гарантує плавність інтерфейсу та зручність використання додатка [5].

Для збереження та ефективного оперування геопросторовими даними об'єктів у ролі системи управління базами даних обрано PostgreSQL із розширенням PostGIS. PostgreSQL є об'єктно-реляційною системою управління базами даних з відкритим вихідним кодом, що підтримує стандарт SQL та забезпечує високий рівень надійності, цілісності даних і масштабованості. Вона орієнтована на роботу зі складними структурами даних та великими навантаженнями, пропонуючи розширені можливості для

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

розробників серверних систем. Система гарантує повну надійність та відповідність принципам ACID, що забезпечує цілісність даних навіть при критичних збоях і є критично важливим для безпечного ведення фінансових транзакцій чи обліку замовлень.

Висока продуктивність індексування досягається завдяки підтримці різноманітних типів індексів, зокрема GIST та B-tree, які надають миттєвий доступ до інформації навіть при великій кількості записів у базі. Також PostgreSQL демонструє гнучкість у роботі з різними типами даних, адже крім стандартних таблиць вона ефективно обробляє формат JSONB, дозволяючи гармонійно поєднувати реляційну структуру з неструктурованими даними без втрати швидкості. Продумана архітектура забезпечує чудову масштабованість, завдяки якій система стабільно працює при зростанні кількості одночасних підключень, що вкрай важливо для платформ із великою аудиторією активних користувачів.

PostGIS додає підтримку географічних об'єктів у SQL і дозволяє виконувати складні просторові запити на рівні бази даних. Це забезпечує високу швидкість обчислення відстаней між користувачами, фільтрацію оголошень у заданому радіусі за допомогою просторових індексів, а також динамічну кластеризацію точок робіт на інтерактивній мапі [6].

Для збереження та відображення зображень користувачів було використано хмарне сховище Azure Blob Storage. Цей інструмент дозволяє повністю винести збереження статичних медіафайлів за межі основної реляційної бази даних, що запобігає її надмірному розростанню та знижує навантаження на сервер [7].

Для підвищення продуктивності, зниження часу відгуку системи до технологічного стека включено сховище даних у оперативній пам'яті Redis. Redis використовується як ефективний механізм кешування частих запитів, що дозволяє уникнути повторного навантаження на базу даних [8].

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Мінімальні вимоги для розгортання системи:

Серверна частина системи потребує двоядерного процесора з архітектурою x64 або ARM64 та тактовою частотою від 2.0 ГГц, а також не менше 4 Гігабайт оперативної пам'яті для одночасної роботи середовища .NET 10, баз даних PostgreSQL з розширенням PostGIS та сховища Redis, Azure Blob Storage. Для розміщення інфраструктури необхідно забезпечити щонайменше 20 Гігабайт постійної пам'яті та стабільне мережеве з'єднання із пропускнуою здатністю від 100 Мегабіт за секунду. У ролі операційної системи може виступати Linux версії Ubuntu 22.04 і вище або Windows Server, обов'язково з підтримкою інструментів Docker та Docker Compose.

Висновки до першого розділу

У першому розділі поставлено задачу на розробку платформи для швидкого пошуку виконавців та замовників короточасної роботи на основі геолокаційних даних, визначено цілі й обмеження системи, а також сформульовано детальні функціональні та нефункціональні вимоги до її базових модулів.

Виконано порівняльний аналіз існуючих програмних аналогів Kabanchik.ua, OLX (Робота) та Upwork, визначено їх переваги, недоліки та архітектурні обмеження, що обґрунтовує доцільність розробки власного сервісу.

Було проведено порівняльний аналіз традиційної монолітної та мікросервісної архітектури й обрано модульну монолітну архітектуру як найбільш оптимальний підхід для реалізації системи.

На основі аналізу предметної області та вимог до продуктивності обґрунтовано вибір інструментальних засобів реалізації. Для серверної частини обрано платформу .NET 10 із технологією ASP.NET Core Minimal APIs та інструментами оркестрації .NET Aspire, для клієнтської частини було обрано фреймворк React Native Expo для кросплатформеної мобільної розробки. Як систему управління базами даних PostgreSQL із просторовим

розширенням PostGIS використано Redis для кешування та Azure Blob Storage для збереження медіафайлів.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ІНТЕРАКТИВНОЇ МАПИ ДЛЯ ПОШУКУ ТА ПРОПОЗИЦІЇ РОБОТИ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

На етапі проєктування платформи для швидкого пошуку короточасної роботи на основі геолокаційних даних було визначено основні варіанти використання програми, що детально описують взаємодію користувачів із системою. Варіантів використання формувалися на основі предметної області та визначених функціональних вимог до додатка.

Архітектура системи передбачає чітке розмежування прав доступу та сценаріїв взаємодії. Клієнт – це користувач, який створює та публікує оголошення про роботи із прив'язкою до географічних координат на інтерактивній мапі, приймає заявки, створює та оплачує контракти. Робітник – це користувач, який займається пошуком робіт, надсилає заявки та виконує роботу за погодженим контрактом з клієнтом.

Перелік варіантів використання системи клієнтом зображено на діаграмі прецедентів (рис. 2.1).

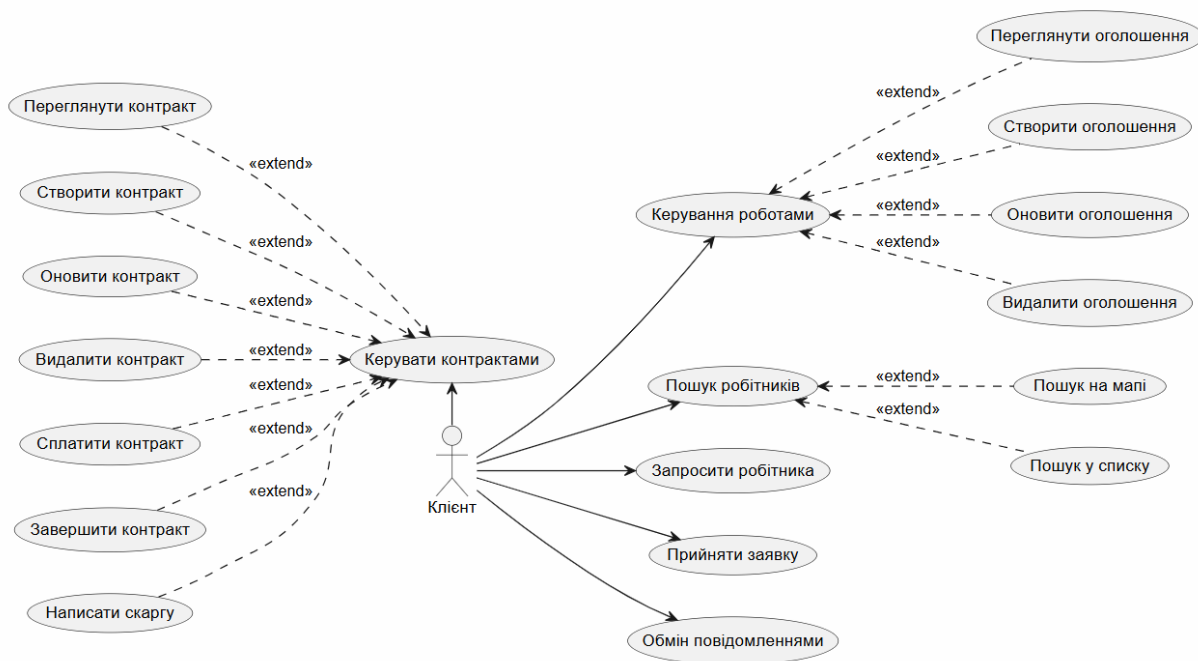


Рис. 2.1. Варіанти використання системи клієнтом.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Перелік варіантів використання системи клієнтом зображено на діаграмі прецедентів (рис. 2.2).



Рис. 2.2. Варіанти використання системи робітником.

Взаємодія користувачів на платформі розпочинається з етапу публікації оголошення та географічного пошуку. Замовник, якому необхідно виконати роботу створює нове оголошення у системі, де детально описує суть праці, визначає бюджет, часові межі та позначає точне місце виконання на інтерактивній мапі. Паралельно з цим робітник, який шукає підробіток, запускає пошук робіт, застосовуючи фільтрацію за радіусом віддаленості від свого поточного місцезнаходження. Система автоматично відсікає всі занадто далекі пропозиції, відображаючи лише ті, що підходять під критерії. Після цього сторони знаходять одна одну: або робітник самостійно створює заявку на вподобане замовлення, а замовник її приймає, або замовник сам знаходить виконавця поблизу і надсилає йому пряме запрошення. Одразу після цього користувачі переходять у внутрішній месенджер, де обговорюють усі деталі та фіксують неформальні домовленості.

Наступний етап охоплює безпосереднє укладання угоди та фіксацію обов'язків. На основі проведених перемовин замовник формує контракт у

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

системі, деталізуючи фінальні вимоги та суму оплати. Робітник переглядає створений контракт та підтверджує його. Щоб гарантувати фінансову безпеку та виключити ризик шахрайства, замовник здійснює оплату контракту, внаслідок чого кошти з його банківської картки списуються та заморожуються на спеціальному рахунку за допомогою інтеграції з платіжними інструментами Монобанк [9]. Після завершення фізичної праці замовник перевіряє результат і, якщо все виконано належним чином, підтверджує успішне завершення контракту. У випадку виникнення конфлікту або недобросовісного виконання обов'язків замовник має можливість подати скаргу, що передає справу на розгляд адміністратором платформи.

Після підтвердження замовником успішного виконання роботи, система автоматично перераховує кошти на внутрішній баланс виконавця. Робітник має можливість у будь-який момент переглянути стан свого рахунку та детальну історію транзакцій, щоб переконатися в отриманні оплати.

Визначений набір прецедентів окреслює сутності, з якими працюватиме система. Загальну структуру предметної області відображено на діаграмі класів (рис. 2.3). Вона описує вимоги до доменної моделі незалежно від технологій чи конкретної реалізації.

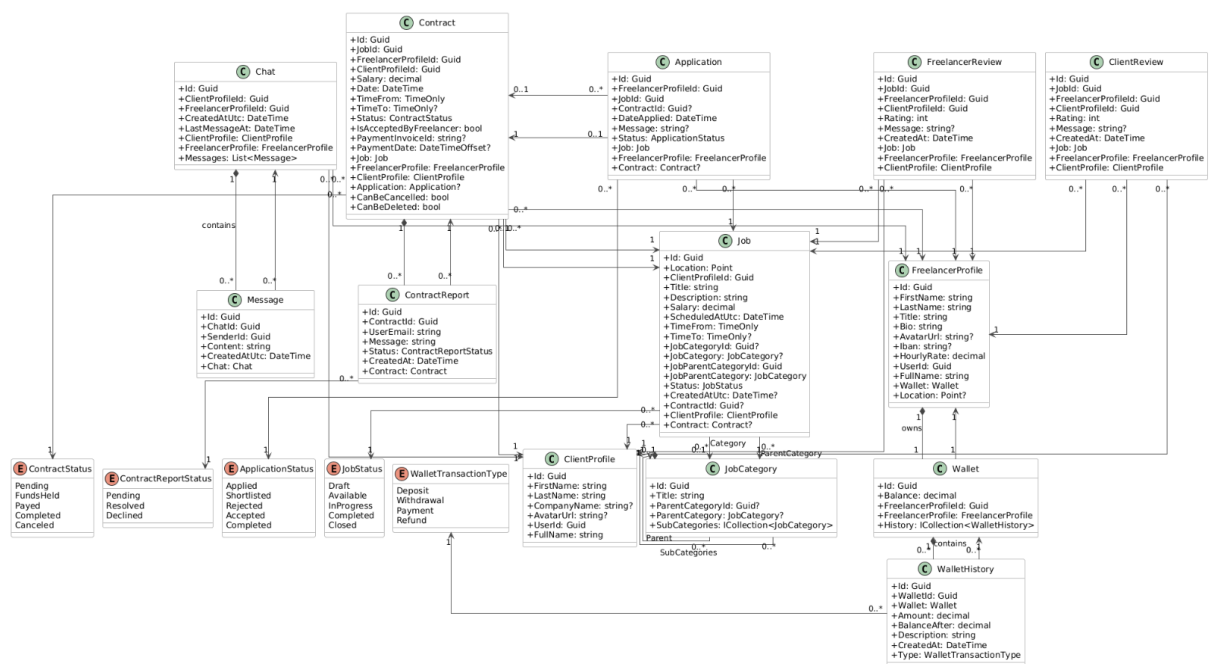


Рис. 2.3. Модель предметної області.

Архітектурна вимога щодо чіткого розділення ролей реалізована через створення двох окремих сутностей для профілів користувачів, які прив'язані до єдиного ідентифікатора облікового запису. Профіль замовника містить персональні дані, назву компанії за наявності. Профіль виконавця, окрім базових персональних даних, містить професійний заголовок, біографію, погодинну ставку та банківські реквізити у форматі IBAN для виведення зароблених грошей. Важливою особливістю профілю виконавця є збереження його геолокації у вигляді просторової точки, що дозволяє системі відображати робітника на мапі для замовників в бажаній локації [10].

Сутність оголошення містить повний опис завдання, пропонований бюджет, часові межі виконання, а також точну геолокаційну мітку на карті, де мають проводитися роботи. Оголошення обов'язково посилається на профіль свого замовника, поточний статус із переліку та відповідну категорію, виступаючи центральним об'єктом навколо якого будується подальша взаємодія сторін.

Процес погодження умов та укладання контракту реалізовано за допомогою сутностей заявок та контрактів. Заявка фіксує відгук конкретного робітника на оголошення, зберігаючи дату подачі, супровідне повідомлення та поточний статус розгляду. Після того, як замовник приймає заявку або самостійно затверджує виконавця, на основі фінальних перемовин створюється контракт. Сутність контракту є ключовим інструментом угоди, оскільки вона фіксує остаточну суму оплати, часові рамки, підтвердження згоди з боку робітника, а також зберігає ідентифікатор інвойсу Monobank та дати платежу, отримані внаслідок оплати контракту. Для вирішення суперечок до контракту може бути прив'язана сутність скарги, яка містить опис проблеми, контактну пошту заявника та статус розгляду конфлікту адміністратором.

Базовим елементом контролю бізнес-процесів у системі є набір переліків, що фіксують стани ключових об'єктів. Перелік статусів оголошення відображає життєвий цикл завдання від публікації до його виконання або

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

закриття. Логіка контрактів та фінансових розрахунків контролюється через перелік статусів контракту, який фіксує моменти очікування, замороження коштів, виплати виконавцю та можливого скасування угоди. Окремі переліки передбачені для моніторингу етапів розгляду скарг арбітражем, керування станами заявок кандидатів у процесі відбору, а також для класифікації типів фінансових транзакцій у внутрішньому гаманці, таких як поповнення, виведення коштів, оплата та повернення.

Внутрішня комунікація та фіксація неформальних домовленостей забезпечується чатами, що дозволяє користувачам не залишати межі захищеного середовища платформи. Сутність чату виступає шляхом для перемовин, що створюється між конкретним замовником та виконавцем.

Після успішного закриття контракту система передбачає створення двох взаємних сутностей: відгуку про замовника та відгуку про виконавця. Обидві сутності містять виставлену оцінку у вигляді числового рейтингу, коментар та дату створення. Це дозволяє накопичувати публічну статистику надійності, що допомагає замовникам обирати потенційних робітників та робітникам просувати власні профілі.

2.2. Розробка бази даних системи

Для забезпечення збереження та обробки даних у системі пошуку та пропозиції короткочасної роботи на основі геолокаційних даних було спроектовано реляційну базу даних, структура якої відображає основні бізнес-процеси платформи. База даних реалізує зберігання інформації про замовників та виконавців, оголошення про фізичні роботи, геолокаційні координати завдань, створені контракти, відгуки та рейтинги, платежі, внутрішні повідомлення чату, а також дані, пов'язані з автентифікацією та ролями доступу користувачів. База даних складається з декількох схем, що відповідають модулям системи [11].

Проектування схеми бізнес-модуля виконувалося з урахуванням принципів нормалізації, що дозволяє зменшити надлишковість даних та забезпечити цілісність інформації. Кожна предметна сутність подається

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

окремою таблицею, а зв'язки між ними реалізовано через зовнішні ключі для підтримання референційної цілісності на рівні системи керування базою даних PostgreSQL з розширенням PostGIS. Загальну структуру спроектованої схеми бази даних бізнес модулю системи подано на рис. 2.4.

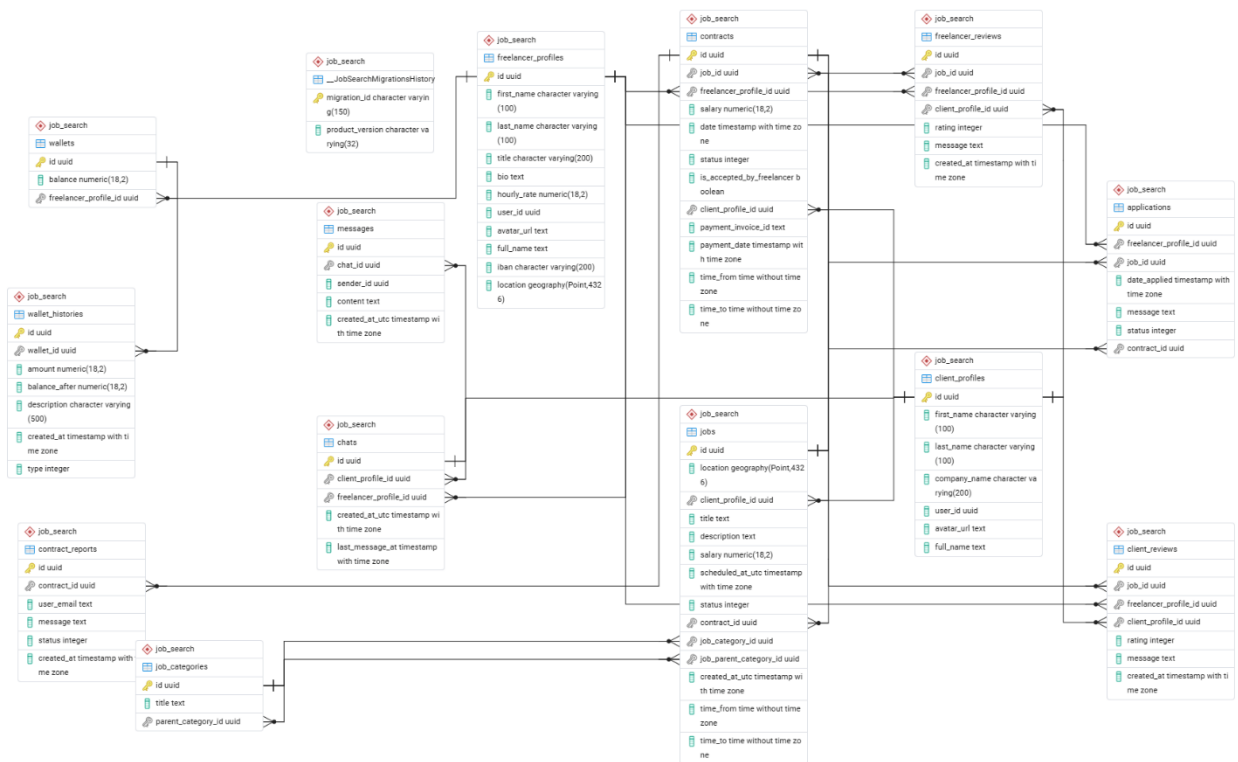


Рис. 2.4. Структура таблиць бази даних схеми бізнес модулю

Було спроектовано схему бази даних для зберігання та обробки інформації про зареєстрованих користувачів системи. Загальну структуру спроектованої схеми подано на рис. 2.5.

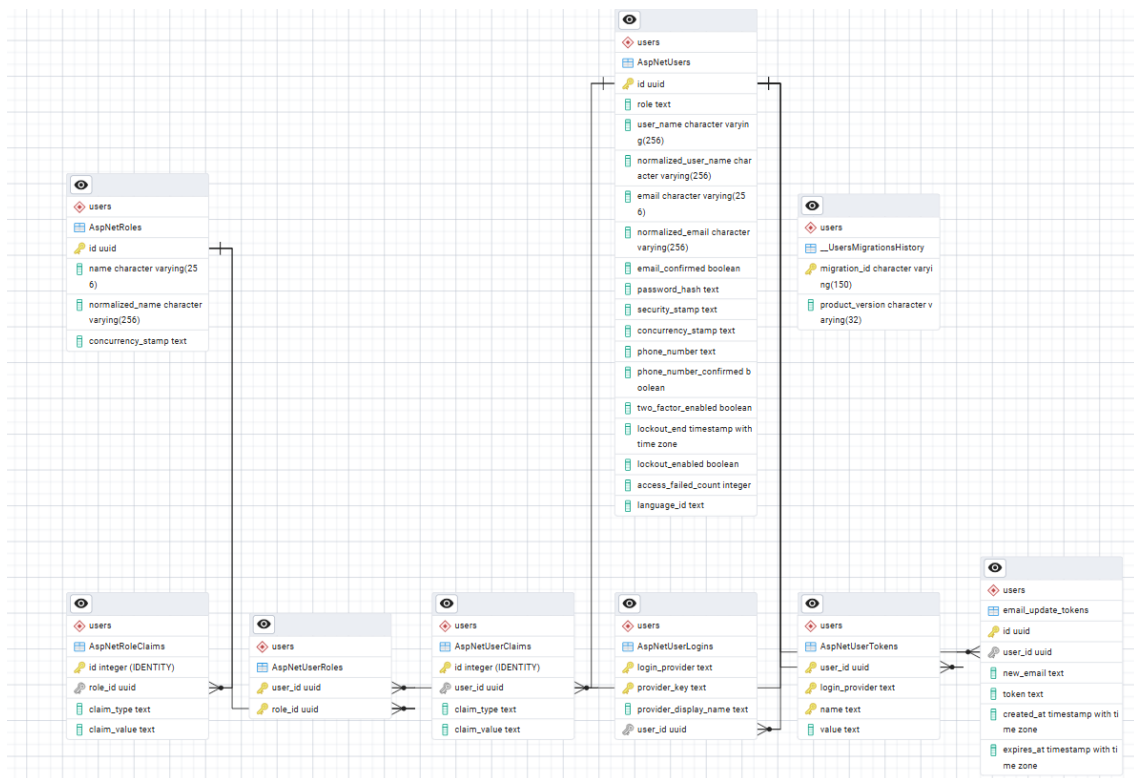


Рис. 2.5. Структура таблиц бази даних схеми модуля користувачів

Було спроектовано схему бази даних для зберігання та обробки інформації про адміністраторів системи, з підтримкою аудит логування дій адміністраторів. Загальну структуру спроектованої схеми подано на рис. 2.6.

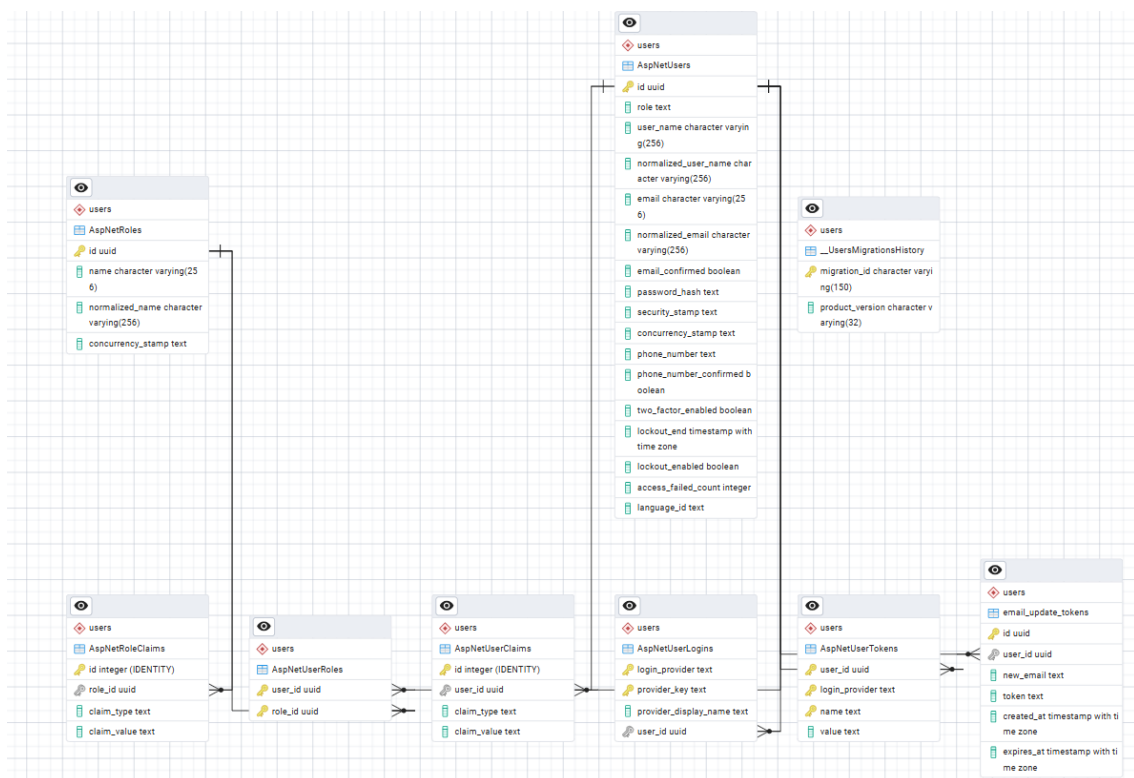


Рис. 2.6. Структура таблиц бази даних схеми модуля адміністрування

Було спроектовано схему бази даних для зберігання та асинхронної обробки подій. Система зберігає події, створені різними модулями, та по чергово обробляє події з підтримкою планування виконання [12]. Загальну структуру спроектованої схеми подано на рис. 2.7.

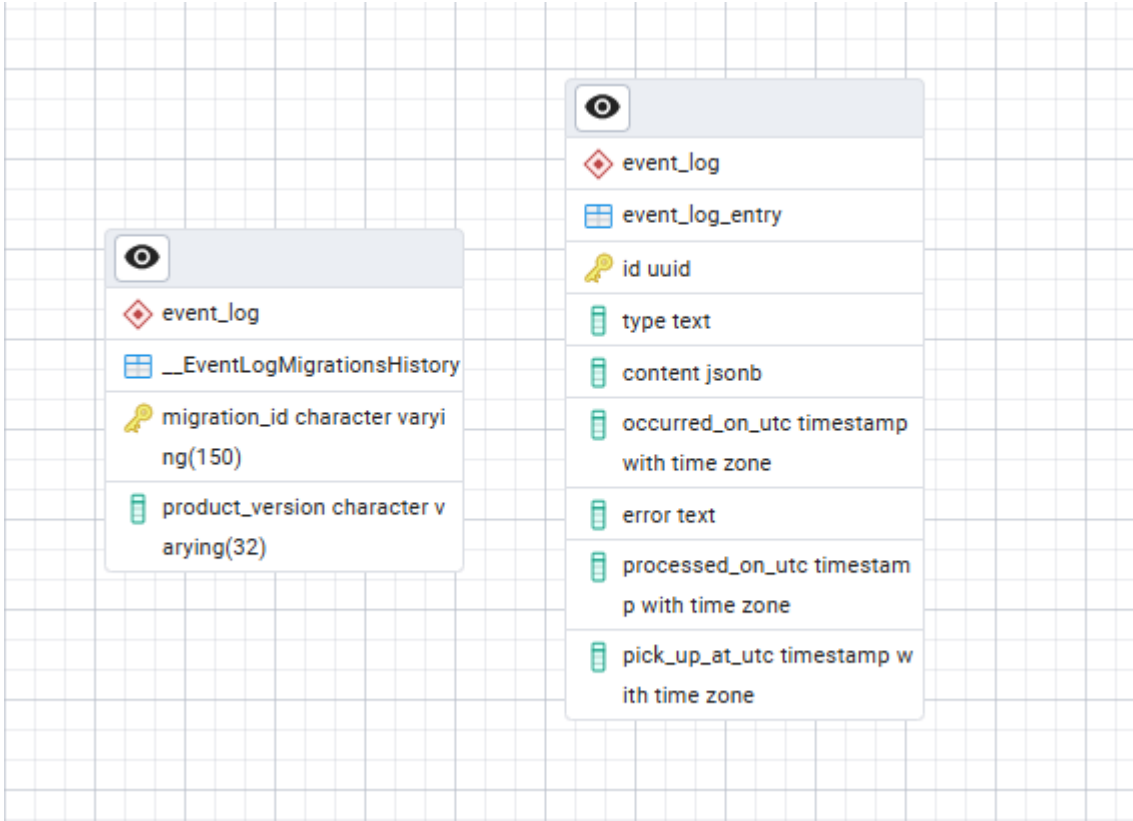


Рис. 2.7. Структура таблиць бази даних схеми модулю подій

Основними елементами реляційної структури даних є профілі користувачів, які розділяються на дві ключові ролі: клієнти та робітники. Детальний опис структури полів та типів даних для зберігання інформації про замовників представлено в табл. 2.1.

Таблиця 2.1

Опис таблиці client_profiles

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор профілю клієнта.

продовж. табл. 2.1

first_name	character varying(100)			Ім'я клієнта.
last_name	character varying(100)			Прізвище клієнта.
company_name	character varying(200)			Назва компанії чи організації (необов'язково).
user_id	uuid			Ідентифікатор облікового запису користувача (зв'язок із таблицею користувачів).
avatar_url	text			Шлях до зображення профілю в хмарному сховищі.
full_name	text			Обчислюване та збережене повне ім'я (first_name + last_name).

У системі функціонують облікові записи виконавців. Вони містять додаткові атрибути, такі як базова погодинна ставка та біографія для формування пошукової видачі майстрів. Структуру відповідної сутності супроводжує детальне виокремлення атрибутів, що наведені в табл. 2.2.

Таблиця 2.2

Опис таблиці freelancer_profiles

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор профілю виконавця.

продовж. табл. 2.2

first_name	character varying(100)			Ім'я виконавця.
last_name	character varying(100)			Прізвище виконавця.
title	character varying(200)			Професійний заголовок або коротка спеціалізація робітника.
bio	text			Детальний опис досвіду та навичок робітника.
hourly_rate	numeric(18,2)			Базова погодинна ставка оплати праці.
user_id	uuid			Ідентифікатор облікового запису користувача.

Присутні сутності оголошення про роботу, які створюються клієнтами та містять просторові координати для відображення на інтерактивній мапі платформи. Взаємозв'язок між замовником та сутністю роботи реалізовано через зовнішній ключ `client_profile_id`. Повний склад полів оголошення описано в табл. 2.3.

Таблиця 2.3

Опис таблиці jobs

Назва	Тип даних	РК	ФК	Опис
id	uuid	+		Унікальний ідентифікатор оголошення про роботу.

продовж. табл. 2.3

location	geography(Point,4326)			Геолокаційна точка на мапі.
client_profile_id	uuid		+	Ідентифікатор профілю клієнта, який створив завдання.
title	text			Назва оголошення (короткий заголовок суті завдання).
description	text			Розгорнутий опис фізичних робіт та вимог до виконавця.
salary	numeric(18,2)			Запропонована сума винагороди.
scheduled_at_utc	timestamp with time zone			Запланована дата та час початку виконання роботи.

Для того, щоб робітник міг запропонувати свої послуги для вирішення опублікованого завдання, у системі передбачено механізм заявок. Заявка пов'язує конкретного робітника із оголошенням, фіксуючи дату відгуку та супровідне повідомлення, що відображено у структурі даних, наведеній у табл. 2.4.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Таблиця 2.4

Опис таблиці applications

Назва	Тип даних	PK	FK	Опис
id	uuid	+		Унікальний ідентифікатор заявки на роботу.
freelancer_profile_id	uuid		+	Ідентифікатор робітника, який подав заявку на виконання.
job_id	uuid		+	Ідентифікатор оголошення про роботу, на яке зроблено відгук.
date_applied	timestamp with time zone			Точна дата та час подання відгуку.
message	text			Супровідне повідомлення від виконавця до клієнта.
status	integer			Стан розгляду заявки
contract_id	uuid		+	Створений контракт

Після погодження умов між сторонами та затвердження заявки у системі фіксується фінансова угода — контракт, що переводить кошти замовника в

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

режим холдування (ескроу) через API Monobank. Параметри контролю життєвого циклу цієї короткострокової угоди структуровано в табл. 2.5.

Таблиця 2.5

Опис таблиці contracts

Назва	Тип даних	PK	FK	Опис
id	uuid	+		Унікальний ідентифікатор контракту короткострокової угоди.
job_id	uuid		+	Посилання на конкретне завдання, за яким укладається договір.
freelancer_profile_id	uuid		+	Ідентифікатор затвердженого виконавця робіт.
client_profile_id	uuid		+	Ідентифікатор клієнта послуг.
salary	numeric(18,2)			Фінально узгоджена сума оплати, яка підлягає холдуванню.
date	timestamp with time zone			Дата та час фіксації та створення контракту.
status	integer			Етап життєвого циклу угоди.

Для ведення обліку зароблених коштів та забезпечення можливості їх подальшого виведення на реальні банківські рахунки, кожен робітник володіє внутрішнім рахунком. Структуру полів цього інструменту наведено в табл. 2.6.

Таблиця 2.6

Опис таблиці wallets

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор внутрішнього рахунку.
balance	numeric(18,2)			Поточний доступний фінансовий баланс робітника на платформі.
freelancer_profile_id	uuid		+	Ідентифікатор профілю робітника, якому належить гаманець.

З метою забезпечення прозорості фінансових розрахунків, захисту від збоїв та ведення обліку, будь-який рух коштів логується в окремому журналі транзакцій. Повний перелік полів для історії гаманцю містить табл. 2.7.

Таблиця 2.7

Опис таблиці wallet_histories

Назва	Тип даних	РК	ФК	Опис
id	uuid	+		Унікальний ідентифікатор логу фінансової операції.
wallet_id	uuid		+	Посилання на гаманець, по якому відбувся рух коштів.
amount	numeric(18,2)			Сума фінансової операції.
balance_after	numeric(18,2)			Залишок грошових коштів на рахунку після проведення поточної операції.
description	character varying(500)			Текстове пояснення призначення транзакції.
created_at	timestamp with time zone			Дата та час проведення фінансової проводки.
type	integer			Тип операції.

Після успішного завершення фізичних робіт за контрактом замовник має можливість оцінити якість послуг виконавця. Дані числового рейтингу та текстового коментаря накопичують публічну статистику надійності фрілансера, як продемонстровано в табл. 2.8.

Таблиця 2.8

Опис таблиці freelancer_reviews

Назва	Тип даних	PK	FK	Опис
id	uuid	+		Унікальний ідентифікатор оцінки/відгуку.
job_id	uuid		+	Ідентифікатор завдання, за результатами якого залишено відгук.
freelancer_profile_id	uuid		+	Профіль робітника, чий рейтинг формується цією оцінкою.
client_profile_id	uuid		+	Профіль клієнта, який є автором відгуку.
rating	integer			Кількісна оцінка якості від 1 до 5.
message	text			Зміст відгуку.
created_at	timestamp with time zone			Дата та час створення відгуку клієнтом.

Відповідно до оцінювання виконавців, архітектура системи передбачає можливість зворотного зв'язку фрілансера щодо адекватності клієнта та умов праці. Це дозволяє підтримувати баланс екосистеми платформи, а структура полів цих відгуків деталізована в табл. 2.9.

Таблиця 2.9

Опис таблиці client_reviews

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор оцінки клієнта.
job_id	uuid		+	Ідентифікатор виконаного завдання.
freelancer_profile_id	uuid		+	Профіль робітника, який є автором відгуку.
client_profile_id	uuid		+	Профіль клієнта, чий рейтинг формується цією оцінкою.
rating	integer			Оцінка клієнта за шкалою від 1 до 5.
message	text			Текстовий коментар від виконавця щодо умов праці та адекватності клієнта.
created_at	timestamp with time zone			Дата та час збереження відгуку.

У разі виникнення конфліктних ситуацій, невиконання зобов'язань або фіксації ознак шахрайства під час виконання активного контракту, будь-яка із сторін може ініціювати спір. Поля сутності наведено в табл. 2.10.

Таблиця 2.10

Опис таблиці contract_reports

Назва	Тип даних	РК	ФК	Опис
id	uuid	+		Унікальний ідентифікатор поданої скарги.
contract_id	uuid		+	Посилання на заблокований контракт, щодо якого виник спір.
user_email	text			Електронна адреса ініціатора скарги (клієнта чи виконавця).
message	text			Текстовий опис суті конфлікту або невиконаних зобов'язань.
status	integer			Статус розгляду скарги внутрішньою службою арбітражу програми.
created_at	timestamp with time zone			Дата та час фіксації претензії в системі.

Для обговорення деталей виконання фізичних робіт та оперативного зв'язку користувачі взаємодіють у внутрішньому месенджері. Первинним елементом цієї підсистеми є кімната діалогу, параметри якої визначено в табл. 2.11.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.11

Опис таблиці chats

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор кімнати чату.
client_profile_id	uuid			Посилання на профіль клієнта – учасника діалогу.
freelancer_profile_id	uuid			Посилання на профіль виконавця – учасника діалогу.
created_at_utc	timestamp with time zone			Дата та час створення діалогу між користувачами.

Безпосередній обмін текстовими повідомленнями всередині створених кімнат діалогів фіксується у базі даних. Структуру сутності повідомлень наведено в табл. 2.12.

Таблиця 2.12

Опис таблиці messages

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор текстового повідомлення.

продовж. табл. 2.12

chat_id	uuid		+	Посилання на ідентифікатор кімнати чату, до якої належить повідомлення.
sender_id	uuid			Ідентифікатор користувача, який є автором повідомлення.
content	text			Текстовий зміст повідомлення.
created_at_utc	timestamp with time zone			Точний час відправлення повідомлення за UTC.

Адміністративна частина платформи вимагає обов'язкового ведення системного аудиту для контролю дій адміністраторів під час вирішення скарг або оновлення даних користувачів. Поля сутності наведено в табл. 2.13.

Таблиця 2.13

Опис таблиці audit_logs

Назва	Тип даних	PK	FK	Опис
id	uuid	+		Унікальний ідентифікатор запису аудиту.

admin_id	uuid		+	Посилання на користувача-адміністратора, який виконав дію.
action	text			Назва системної дії, що логується.
old_values	jsonb			Попередній стан змінених полів об'єкта у форматі JSON.
new_values	jsonb			Новий стан полів об'єкта після оновлення у форматі JSON.
parameters	text[]			Масив текстових параметрів або ідентифікаторів сутностей, що брали участь в операції.
timestamp	timestamp with time zone			Точна дата та час фіксації адміністративної дії.

Для реалізації асинхронної обробки подій і гарантування відмовостійкості системи під час збоїв, усі події спочатку надійно зберігаються у базі даних. Склад полів сутності подій представлено в табл. 2.14.

Таблиця 2.14

Опис таблиці event_log_entry

Назва	Тип даних	РК	FK	Опис
id	uuid	+		Унікальний ідентифікатор системної доменної події.
type	text			Повна кваліфікована назва класу події .NET.
content	jsonb			Серіалізовані в JSON дані події для асинхронної обробки.
occurred_on_utc	timestamp with time zone			Дата та час виникнення події в системі за UTC.
error	text			Стек помилки у разі збою під час обробки події фоновим воркером.
processed_on_utc	timestamp with time zone			Час успішного завершення обробки події.
pick_up_at_utc	timestamp with time zone			Час наступної спроби повторної обробки події у разі попереднього збою.

Описана структура бази даних повністю покриває потреби всіх основних модулів системи. Організація даних ефективно підтримує поточний функціонал платформи і має високий потенціал до масштабування.

2.3. Проєктування та реалізація алгоритмів роботи системи

Поведінка платформи, її внутрішні алгоритми та стани описуються за допомогою трьох типів діаграм: активності, послідовності та станів. Описано основні алгоритми використання платформи.

Створення та виконання роботи є ключовим алгоритмом використання платформи. Взаємодія на платформі розпочинається з того, що клієнт створює та публікує оголошення про роботу, обов'язково вказуючи його географічне розташування на інтерактивній мапі. Паралельно з цим робітник шукає відповідні роботи в системі, використовуючи фільтрацію за радіусом віддаленості. Знайшовши підходящий варіант, робітник подає заявку на обране оголошення. Клієнт отримує цю заявку, приймає її та обговорює з виконавцем усі деталі у внутрішньому месенджері платформи. Після узгодження неформальних домовленостей клієнт створює в системі контракт із фінальними вимогами та сумою оплати. Робітник переглядає умови та підтверджує цей контракт, після чого клієнт здійснює оплату. Система через інтеграцію з Monobank списує кошти з картки клієнта та заморожує їх на спеціальному рахунку для забезпечення фінансової безпеки обох сторін. Отримавши підтвердження безпеки, робітник виконує фізичну працю, а після її завершення клієнт ретельно перевіряє отриманий результат. Якщо роботу виконано якісно, клієнт підтверджує успішне завершення контракту, і система автоматично перераховує заморожені кошти на внутрішній баланс виконавця. У разі виникнення конфлікту або недобросовісного виконання обов'язків клієнт подає скаргу, і вирішенням ситуації починає займатися адміністратор платформи. Наприкінці процесу робітник перевіряє стан свого рахунку та історію транзакцій, щоб переконатися в отриманні виплати. Діаграма активності створення та виконання роботи подана на рис. 2.8.

		<i>Зайцев О.І.</i>			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		<i>Морозов А.В.</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		46

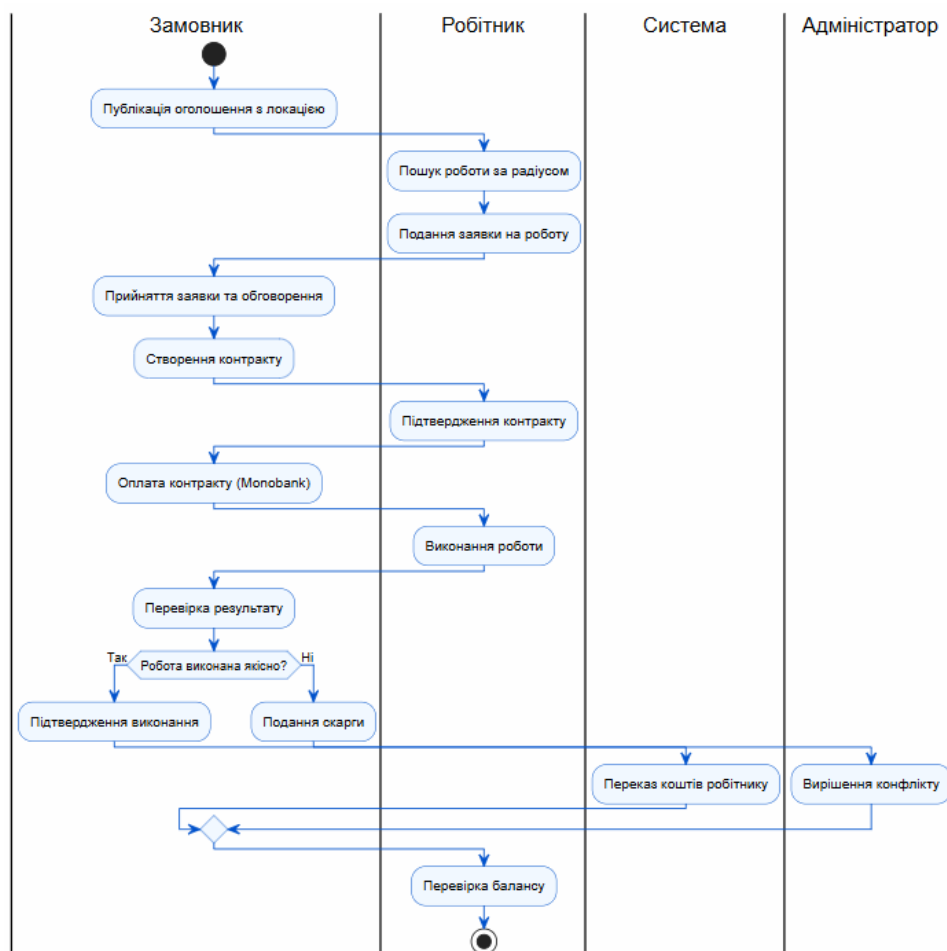


Рис. 2.8. Діаграма активності створення та виконання роботи

Життєвий цикл контракту розпочинається у стані очікування, коли клієнт створює нову угоду в системі. Після того як робітник ознайомлюється з умовами та погоджує контракт, він переходить у стан прийнятого. Наступний етап настає тоді, коли клієнт здійснює оплату, внаслідок чого кошти утримуються платіжною системою, а сам контракт набуває відповідного статусу заморожених грошей. У цьому стані клієнт має право скасувати платіж протягом двадцяти чотирьох годин і повернути контракт назад у початковий стан очікування. Якщо ж протягом доби скасування не відбулося, оплата автоматично фіналізується, і угода переходить у статус активної, що означає початок виконання роботи робітником. Фінал взаємодії залежить від результатів праці та її перевірки. За умови успішного виконання та здачі роботи контракт переводиться в завершений стан. Якщо ж між сторонами виникає конфлікт або фіксується недобросовісне виконання обов'язків, контракт набуває статусу оскаржуваного та передається на розгляд

адміністрації платформи. Обидва ці варіанти є кінцевими точками в життєвому циклі поточної угоди. Діаграма станів контрактів подана на рис. 2.9.



Рис. 2.9. Діаграма станів контрактів

Процес фіналізації контракту складається з двох послідовних етапів, завдяки чому система автоматично контролює час заморожування грошей. Все починається з того, що клієнт підтверджує оплату всередині мобільного додатка Monobank. Після цього банк затискає необхідну суму на картці клієнта, переводячи транзакцію у внутрішній статус утримання коштів. Одразу після цього платіжний шлюз Monobank надсилає асинхронне сповіщення на спеціально виділений платіжний вебхук платформи [13]. Отримавши цей запит, платіжний вебхук самостійно змінює стан контракту на утримання коштів та реєструє в базі даних майбутню подію фіналізації, встановлюючи для неї точний час виконання на кінець наступної доби. Після успішного планування вебхук повертає банку позитивну відповідь про успішне прийняття даних. Другий етап запускається автоматично за розкладом. Внутрішній планувальник задач системи постійно сканує базу даних на наявність відкладених подій, час виконання яких уже настав.

Виявивши заплановану раніше подію, планувальник підхоплює її та передає на опрацювання в модуль контрактів. Цей модуль виконує фінальну бізнес-логіку, переводячи статус контракту в активний, що є зеленим світлом для початку виконання робіт робітником. Наприкінці система звітує перед планувальником про успішне закриття задачі, і процес фіналізації завершується. Діаграма послідовності для оплати контракту подана на рис. 2.10.

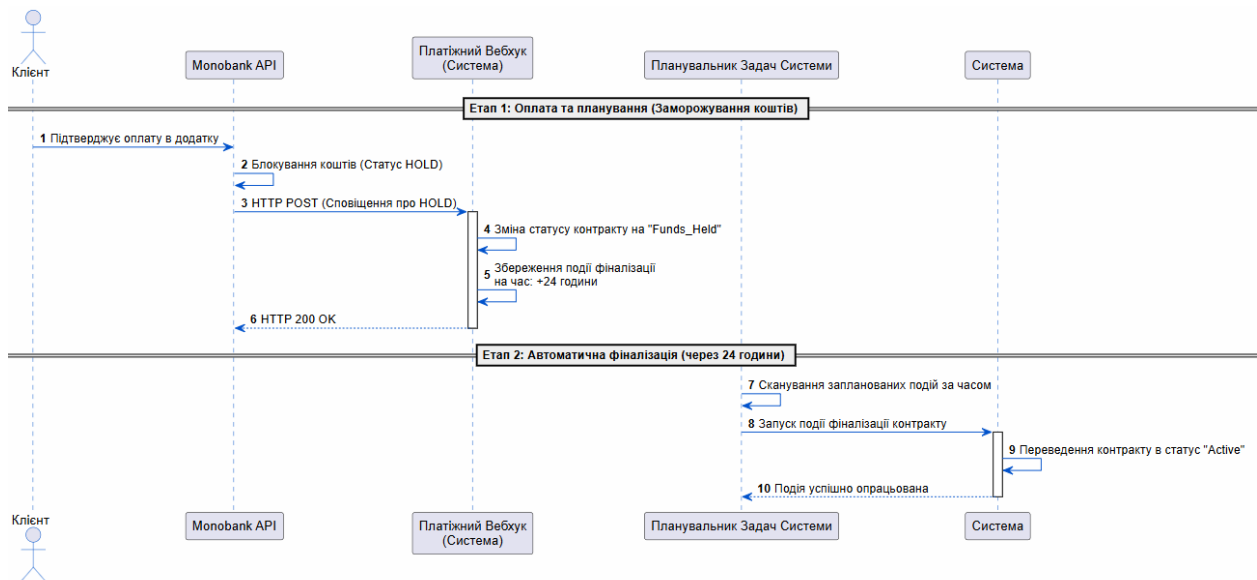


Рис. 2.10. Діаграма послідовності для оплати контракту

Процес реєстрації користувача розпочинається з того, що користувач вводить адресу електронної пошти, пароль та одразу обирає свою майбутню роль клієнта або робітника. Система обробляє ці дані, фіксує внутрішню подію первинної реєстрації та надсилає верифікаційний лист [28]. Коли користувач відкриває пошту та переходить за посиланням, система реєструє подію підтвердження пошти, ініціює надсилання автоматичного вітального листа та виконує вхід до облікового запису. На фінальному етапі користувач, який уже увійшов у систему, заповнює детальний профіль відповідно до обраної ролі. Система зберігає ці дані, створює та активує особистий профіль, після чого відкриває користувачу повний доступ до його робочого кабінету. Діаграма послідовності реєстрації створення та виконання роботи подана на рис. 2.11.

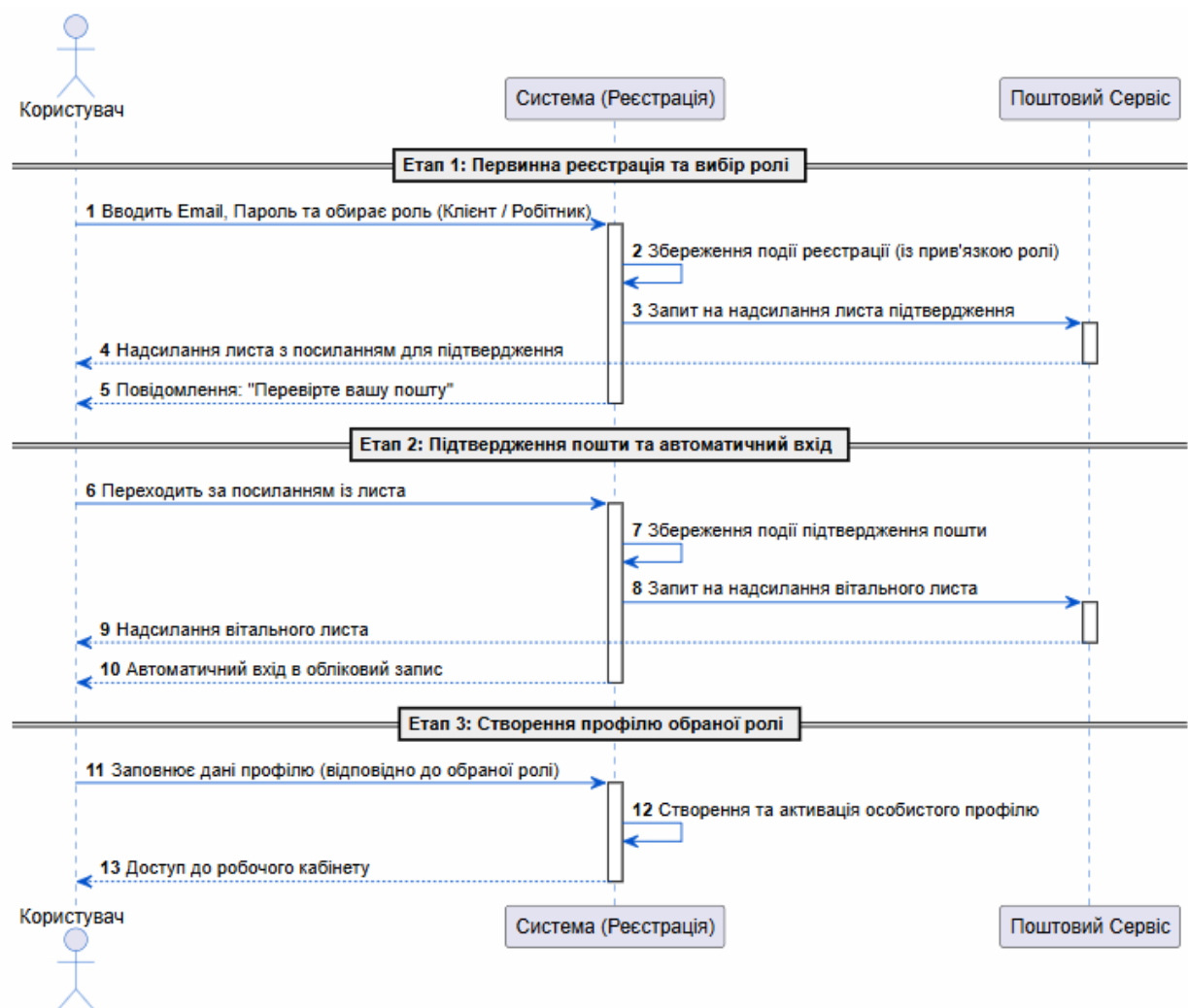


Рис. 2.11. Діаграма послідовності реєстрації

2.4. Реалізація функціоналу

Важливим елементом функціоналу сервісу інтерактивна мапа для пошуку та пропозиції роботи є процес ефективного завантаження та кешування геопросторових даних робіт для їх подальшого відображення на інтерактивній мапі [14]. Оскільки робота з фрагментами мапи вимагає високої швидкості обробки великої кількості запитів, реалізацію архітектурних принципів оптимізації та взаємодії з кешем доцільно розглянути на прикладі ендпоінту отримання зведеної JSON-інформації про роботи для конкретного фрагмента мапи.

```

public void MapEndpoint(IEndpointRouteBuilder app)
{
    app.MapGet("jobs/tiles/summary/{z:int}/{x:int}/{y:int}.json", HandleAsync)
        .WithTags("Jobs")
        .WithName("GetJobSummaryJsonTile");
}
  
```

Реалізація цього сценарію побудована на чіткому розподілі обов'язків між різними компонентами програми. Маршрутизація відповідає за приймання зовнішнього запиту, окрема перевірка контролює правильність вхідних координат, а інструменти роботи з кешем та базою даних забезпечують швидкий доступ до інформації. Завдяки такій структурі вся логіка вибірки та об'єднання даних про роботи схована всередині репозиторію.

На рівні коду взаємодія з користувачем налаштована через технологію Minimal API платформи .NET. Створено окремий клас GetJobSummaryJsonTile. Метод MapEndpoint реєструє в системі правильну адресу запиту та додає мітку "Jobs" для зручного групування в документації.

```
private static async Task<IResult> HandleAsync(
    int z, int x, int y,
    IJobRepository jobRepository,
    ICacheService cacheService,
    HttpContext httpContext,
    CancellationToken ct)
{
    if (!TileHelper.IsValid(z, x, y, out var error))
        return Results.BadRequest(error);

    var cacheKey = $"job-tile:summary:json:{z}:{x}:{y}";
    var points = await cacheService.GetOrSetAsync(
        cacheKey,
        () => jobRepository.GetJobSummaryJsonTileAsync(z, x, y, ct),
        new DistributedCacheEntryOptions
        {
            AbsoluteExpirationRelativeToNow = z < 13 ? TimeSpan.FromMinutes(10) : TimeSpan.FromSeconds(30)
        },
        ct);

    return await TileHelper.CompressedResultAsync(
        points,
        "application/json",
        httpContext,
        ct);
}
```

Сам ендпоінт не шукає дані в базі, а працює як диспетчер у методі HandleAsync. Спочатку за допомогою інструменту TileHelper.IsValid програма перевіряє, чи існують взагалі передані координати фрагмента мапи. Якщо координати помилкові, запит одразу скасовується із помилкою. Якщо все добре, щоб не перевантажувати базу даних, система створює унікальний ключ для кешу на основі координат. Далі сервіс кешування перевіряє, чи не запитував хтось цей фрагмент раніше. Якщо дані вже є в кеші, вони

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

повертаються клієнту миттєво. Якщо даних немає, програма звертається до репозиторію робіт, завантажує їх із бази та зберігає в кеш. Час, на який дані залишаються в кеші, налаштовується автоматично: для загальних великих планів мапи, де інформація змінюється рідко, дані зберігаються довше (10 хвилин), а для детальних дрібних вулиць, де важлива точність, кеш оновлюється кожні 30 секунд. Наприкінці оброблені дані передаються у метод `TileHelper.CompressedResultAsync`, який просто формує правильну JSON-відповідь і віддає її користувачу.

Важливим елементом системи є надійна обробка та надсилання внутрішніх подій, які створюються під час роботи програми (патерн Outbox). Оскільки події мають надходити до інших частин системи без збоїв і дублювань, цей процес винесено в окреме фонове завдання, яке запускається автоматично за розкладом. Розглянемо, як влаштована ця логіка, на прикладі класу `ProcessEventLogsJob`.

```
[DisallowConcurrentExecution]
public class ProcessEventLogsJob(
    IEventLogService outboxMessageService,
    IUnitOfWork unitOfWork,
    IPublisher publisher,
    IOptions<EventLogOptions> options,
    ILogger<ProcessEventLogsJob> logger) : IJob
```

Реалізація цього завдання побудована на чіткому розподілі обов'язків. Атрибут `DisallowConcurrentExecution` на самому початку гарантує, що якщо попередній запуск завдання ще не завершився, новий не розпочнеться — це захищає систему від повторної обробки тих самих записів. Головний метод `Execute` виконує роль координатора: він завантажує порцію необроблених подій, перетворює їх на зрозумілий для системи формат, викликає інструмент для їхнього розсилання та фіксує результат у базі даних. Вся логіка збереження історії подій прихована в сервісі журналів, а безпосереднє відправлення покладається на внутрішній медіатор.

На рівні коду завдання реалізовано як клас `ProcessEventLogsJob`, який впроваджує інтерфейс `IJob` бібліотеки `Quartz.NET`. Через конструктор клас отримує всі необхідні для роботи інструменти [15].

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Під час запуску фонового завдання метод `Execute` насамперед звертається до сервісу подій `outboxMessageService`, щоб отримати список ще не оброблених записів. Кількість записів, які беруться за один раз, чітко обмежена налаштуваннями програми (параметр `BatchSize`), щоб не перевантажувати пам'ять. Якщо нових подій немає, завдання одразу завершує свою роботу.

```
public async Task Execute(IJobExecutionContext context)
{
    var entries = await outboxMessageService.GetUnprocessedEvents(options.Value.BatchSize,
context.CancellationToken);

    if (entries.Count == 0) return;

    foreach (var entry in entries)
    {
        InternalEvent? @event = entry.DeserializeContent();

        if (@event is null)
        {
            logger.LogWarning("Could not deserialize event log entry {EntryId}.", entry.Id);
            continue;
        }

        try
        {
            await publisher.Publish(@event, context.CancellationToken);
            entry.ProcessedOnUtc = DateTime.UtcNow;
        }
        catch (Exception ex)
        {
            entry.Error = ex.ToString();
            logger.LogError(ex, "Failed to publish event log entry {EntryId}.", entry.Id);
        }
    }

    await unitOfWork.SaveChangesAsync(context.CancellationToken);
}
```

Якщо ж записи знайдено, програма починає по черзі обробляти кожен із них у циклі. Спочатку текстовий вміст події перетворюється назад у об'єкт коду. Якщо цей процес ламається і подія не розпізнається, система записує попередження в лог і переходить до наступного запису. Для кожної успішно розпізнаної події програма намагається опублікувати її через сервіс `publisher`. Якщо публікація пройшла успішно, запису присвоюється поточний час обробки. Якщо під час надсилання виникає будь-яка помилка, система перехоплює її, записує текст помилки безпосередньо в цей запис для подальшого аналізу та фіксує збій у загальному лозі програми. Наприкінці,

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

після обробки всієї пачки подій, викликається метод `unitOfWork.SaveChangesAsync`, який зберігає всі оновлені статуси та помилки в базу даних.

Критично важливим елементом фінансової безпеки є обробка асинхронних сповіщень від зовнішнього платіжного шлюзу Monobank. Функціонал приймання та логічного опрацювання таких запитів реалізовано у класі `PaymentWebhookMono`, який інтегрується через інтерфейс `IEndpoint`.

```
public void MapEndpoint(IEndpointRouteBuilder app)
{
    app.MapPost("contracts/payment-webhook/mono", HandleAsync)
        .WithTags("Contracts")
        .WithName("PaymentWebhookMono");
}
```

Основне розгалуження логіки життєвого циклу контракту побудоване на аналізі платіжного статусу `payload.Status.ToLowerInvariant()` за допомогою конструкції `switch`. За допомогою сервісу `integrationEventService` сукупні зміни записуються в базу даних разом із реєстрацією інтеграційної події `ContractPayedEvent` для її відкладеної обробки воркером у межах патерну `Transactional Outbox`. У випадку статусу `"failure"`, який обробляє ситуацію невдалої оплати або скасування інвойсу користувачем, посилання на ідентифікатор інвойсу обнуляється, а стан об'єкта синхронізується зі сховищем даних. При отриманні фінального статусу `"success"`, що сигналізує про успішне списання грошових коштів, статус угоди змінюється на `Payed` з остаточною фіксацією змін у реляційній базі даних за допомогою `Unit of Work`.

```
case "success":
    contract.Status = ContractStatus.Payed;
    await unitOfWork.SaveChangesAsync();
    break;
```

При отриманні статусу `"hold"`, який фіксує момент успішного замороження коштів на картці замовника банком-екваєром, система виконує комплекс перевірок поточного стану сутностей. Активувати холдування дозволено лише для угод у статусі `Pending` та доступних оголошень `Available`. Після зміни статусу контракту на `FundsHeld` та переводу роботи в стан

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

виконання InProgress, розраховується часова мітка автоматичної фіналізації платежу.

```
case "hold":
```

```
    if (contract.Status != ContractStatus.Pending)
        break;
```

```
    if (job.Status != JobStatus.Available)
```

```
        return TypedResults.BadRequest("Only available jobs can have contracts activated.");
```

```
    contract.Status = ContractStatus.FundsHeld;
```

```
    job.ContractId = contract.Id;
```

```
    job.Status = JobStatus.InProgress;
```

```
    DateTime nextAvailableMidnight = DateTime.UtcNow.AddDays(2).Date;
```

```
    await integrationEventService.SaveEventAndChangesAsync(new ContractPayedEvent(contract.Id),
nextAvailableMidnight);
```

```
    break;
```

Наприкінці методу шлюзу Monobank повертається успішна відповідь TypedResults.Ok() для підтвердження успішності завершення поточної операції.

Висновки до другого розділу

Було проведено детальний аналіз предметної області платформи швидкого пошуку короточасної роботи на основі геолокаційних даних з позицій об'єктно-орієнтованого підходу. Визначено ключові вимоги до функціонала системи, а також виявлено основних акторів взаємодії, клієнта та робітника, та їхні типові сценарії роботи з платформою.

Для візуалізації структури та логіки додатка було розроблено діаграми прецедентів та класів предметної області. Це дозволило чітко розмежувати ролі користувачів через створення окремих профілів, пов'язаних із єдиним обліковим записом, відокремити сутності оголошень, заявок і контрактів, а також налаштувати зв'язки та архітектурні межі між ними.

На основі визначених доменних сутностей було спроектовано реляційну базу даних у СУБД PostgreSQL із використанням розширення PostGIS для ефективної роботи з просторовими даними. Проєктування бази даних

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

здійснювалося за модульним принципом із рознесенням таблиць по окремих схемах.

Здійснено моделювання поведінки системи за допомогою діаграм активності, станів та послідовності. Було деталізовано та оптимізовано низку критично важливих бізнес-алгоритмів: процес створення та виконання роботи, життєвий цикл контракту, процедуру реєстрації користувачів із верифікацією, а також двоетапний механізм оплати та фіналізації контрактів через платіжні інструменти Monobank.

Реалізовано програмну складову backend-частини системи у вигляді високопродуктивного RESTful API на базі платформи .NET із використанням підходу Minimal APIs. Для забезпечення високої швидкості рендерингу інтерактивної карти впроваджено ендпоінт генерації зведеної JSON-інформації про роботи для конкретних тайлів із застосуванням гнучкого двоярусного механізму кешування (залежно від масштабу мапи) та стиснення результатів.

Для гарантування надійності, асинхронності та цілісності інтеграційних процесів у межах архітектури системи було успішно імплементовано патерн Transactional Outbox. Створено обробку подій на базі бібліотеки Quartz.NET, яка надійно виконує фонову роботу, що унеможливорює втрати даних у разі системних збоїв.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ ІНТЕРАКТИВНОЇ МАПИ ДЛЯ ПОШУКУ ТА ПРОПОЗИЦІЇ РОБОТИ

3.1. Порядок розгортання та налаштування параметрів платформи

Розгортання та налаштування параметрів системи передбачає підготовку серверного середовища, розгортання серверної частини та мобільного додатка, а також налаштування зв'язків між інфраструктурними компонентами. Розроблювальний геоінформаційний сервіс реалізовано на основі модульної монолітної архітектури, де всі домени виконуються всередині єдиного програмного процесу, але суворо ізольовані на рівні коду та схем даних. До складу інфраструктури платформи входить серверна частина на базі .NET 10 із використанням технології ASP.NET Core. Клієнтську функціональність забезпечує кросплатформений мобільний додаток на базі фреймворку React Native та платформи Expo. Для збереження та ефективного оперування геопросторовими даними застосовано реляційну базу даних PostgreSQL із розширенням PostGIS. Також інфраструктура використовує розподілений кеш Redis для зниження часу відгуку системи, хмарне об'єктне сховище Azure Blob Storage для збереження статичних медіафайлів та платіжний шлюз Monobank для реалізації логіки безпечних контрактів із холдуванням коштів. Також реалізовано React додаток для адміністрування платформи, який інтегрується у серверну частину платформи.

Для спрощення розгортання й локального тестування вся конфігурація серверного середовища та керування залежностями описані в коді інструменту оркестрації .NET Aspire. Це дозволяє відтворити та запустити середовище розгортання у будь-який момент часу, забезпечуючи однакову поведінку системи в режимах розробки, тестування й експлуатації.

Розгортання платформи включає кілька послідовних кроків. Спочатку на цільовому серверному середовищі готується базова інфраструктура й операційна система Linux Ubuntu 22.04 або Windows Server із підтримкою Docker та Docker Compose [16]. На наступному етапі вихідний код системи

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

отримується з репозиторію за допомогою системи контролю версій Git. Після цього налаштовуються параметри конфігурації у файлі appsettings.json відповідно до режиму експлуатації. На фінальному кроці виконується збірка та запуск серверних компонентів через .NET Aspire AppHost, який автоматично формує взаємозв'язки між модулями, підключає бази даних та ініціює автоматичне застосування міграцій бази даних PostgreSQL, після чого вебсервер ASP.NET Core починає одночасно обслуговувати як API, так і клієнтські файли React-додатка адміністрування [17].

Для представлення процесу розгортання було наведено діаграму на рис. 3.1.

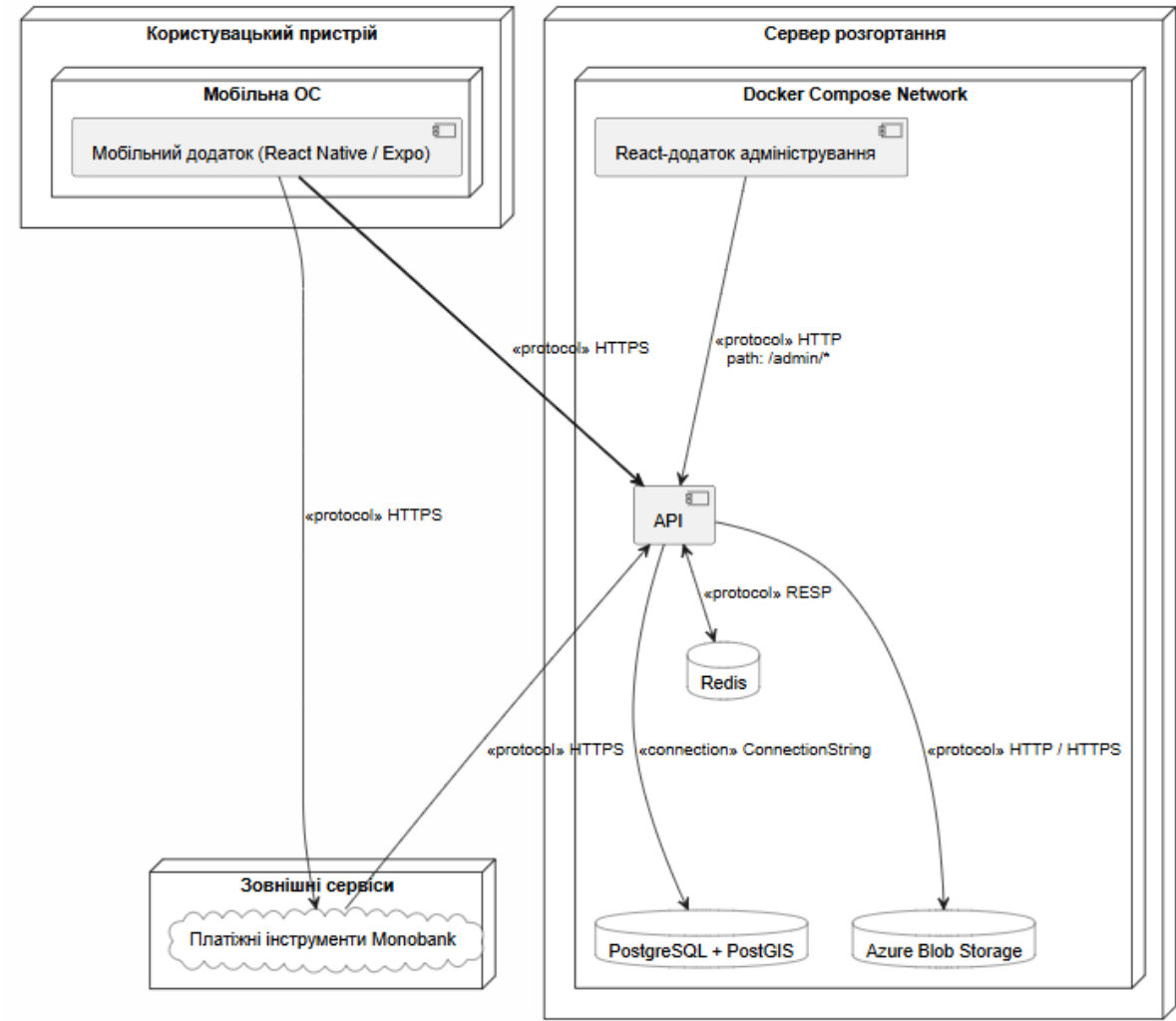


Рис. 3.1. Діграма розгортання

Структура розгортання містить клієнтський мобільний додаток React Native, який взаємодіє з апаратними модулями геолокації смартфона та

відображає інтерактивну мапу. Також структура розгортання містить серверний додаток на .NET 10, який оброблює API запити та обслуговує інтегрований React додаток для адміністрування. Структура включає СУБД PostgreSQL і PostGIS, кеш Redis для швидкої обробки запитів, а також сховище Azure Blob Storage.

Окрім обслуговування HTTP-запитів до API та віддачі статичних файлів адміністративної панелі, серверний .NET-додаток також виступає в ролі фонового процесора. Він бере на себе відповідальність за періодичне сканування, валідацію та послідовне опрацювання черги системних подій у фоновому режимі, що дозволяє ізолювати важкі інтеграційні операції та фінансові розрахунки від основного потоку взаємодії з користувачем.

До параметрів платформи належать рядки підключення до баз даних і сховищ, облікові дані інфраструктурних сервісів, параметри автентифікації користувачів та адміністраторів, ключі доступу до зовнішніх сервісів, а також налаштування платіжної інтеграції та поштової служби. Параметри поділяються на дві категорії, а саме конфігураційні, які містять URL адреси сервісів, налаштування ведення логів, дозволені CORS походження, обмеження таймаутів, інтервалів та розмірів пачок даних. І секретні, до яких належать паролі, токени доступу, ключі шифрування JWT та унікальні API-ключі. Конфігураційні параметри зберігаються у файлах appsettings.json та відрізняються залежно від середовища розгортання [18]. Секретні параметри в реальних умовах експлуатації мають передаватися через змінні середовища операційної системи [19].

Для детальної перевірки та інтерактивного тестування функціональності платформи у режимі розробки автоматично генеруються два окремі незалежні документи специфікації OpenAPI. Перший документ повністю описує публічний інтерфейс для користувацьких сервісів і мобільного додатка, охоплюючи функціонал системи. Другий документ призначений виключно для обслуговування підсистеми внутрішнього адміністрування платформи. Специфікації формуються динамічно на основі метаданих серверної частини,

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

що гарантує повну відповідність документації актуальному стану вихідного коду системи [20].

3.2. Структура інтерфейсу мобільного додатку

Клієнтська частина платформи складається з двох основних інтерфейсів: кросплатформеного мобільного додатка для користувачів та вбудованої вебпанелі для адміністрування системи. Нижче наведено основні екрани інтерфейсу мобільного додатку системи з описом доступних дій користувачів відповідно до їхніх ролей.

Після відкриття додатку, користувач бачить екран входу в обліковий запис з можливістю перейти до реєстрації (рис. 3.2). Поле паролю призовує введене користувачем значення.

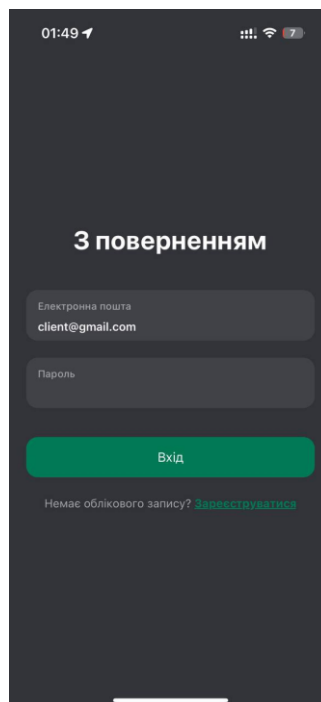


Рис. 3.2. Екран входу

На екрані реєстрації користувач може обрати свою роль відповідно до опису (рис. 3.3). Ролі представлені відповідними картинками та містять додатковий опис. Також можна повернутись до входу за посилання або жестом.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

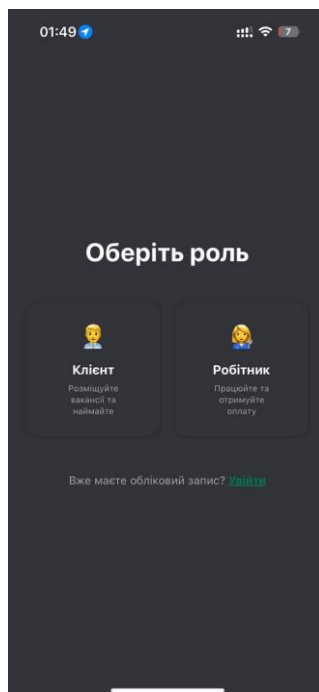


Рис. 3.3. Екран вибору ролі

Після вибору ролі користувач вводить електронну пошту та пароль. Також доступні посилання на сторінку входу та сторінку умов та політики конфіденційності (рис. 3.4) [21].

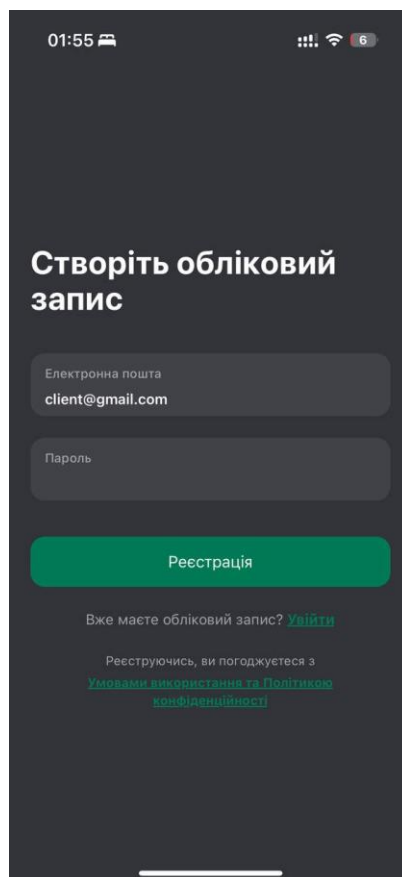


Рис. 3.4. Екран реєстрації

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Після входу в обліковий запис, користувач заповнює власний професійний профіль (рис. 3.5). Клієнти вводять ім'я прізвище, компанію за бажанням.

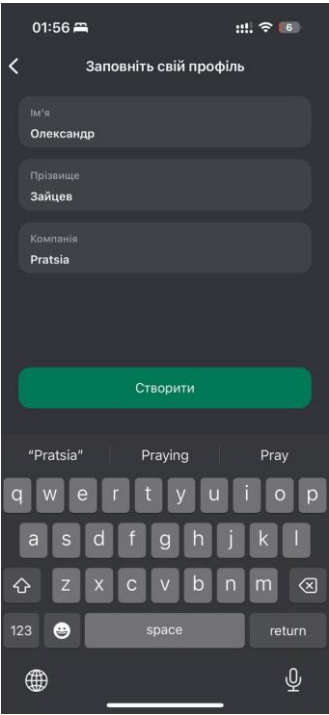


Рис. 3.5. Заповнення профілю клієнта

Робітники вводять ім'я, прізвище, посаду, біографію та погодинну ставку (рис. 3.6). Після заповнення профілю, користувачі мають повний доступ до функціоналу платформи.

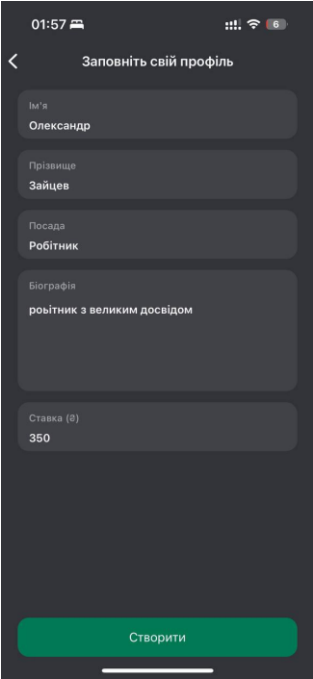


Рис. 3.6. Заповнення профілю робітника

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувач має доступ до вікна власного профілю (рис. 3.7). Це приватна сторінка користувача з меню до всіх необхідних налаштувань додатку.

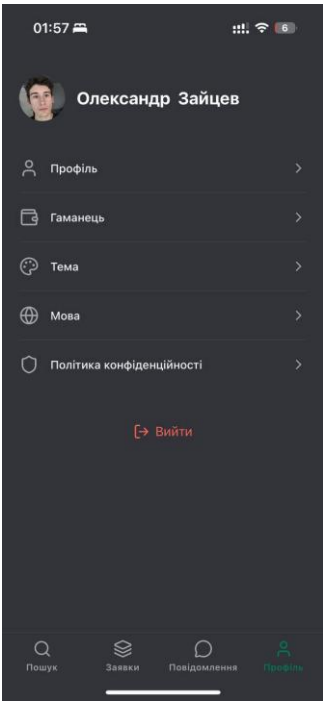


Рис. 3.7. Профіль користувача

Можна змінити тему та мову додатку (рис. 3.8-3.9). Також є опція перегляду політики конфіденційності. Сторінка відкриється в браузері та відобразить актуальний текст.

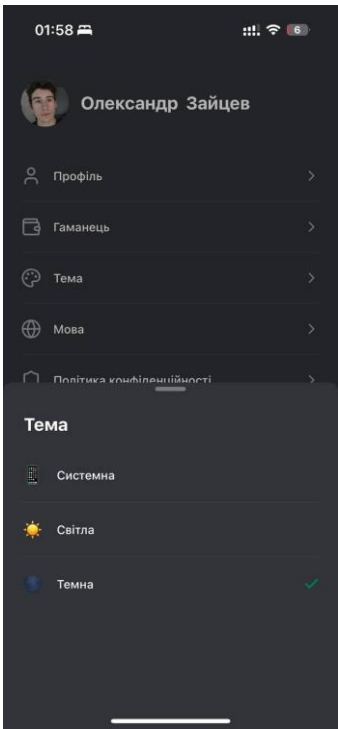


Рис. 3.8. Зміна теми

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

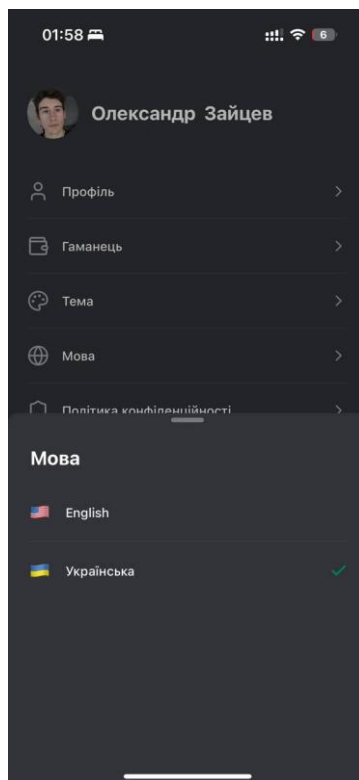


Рис. 3.9. Зміна мови

Користувач може оновлювати власну інформацію на сторінці налаштувань профілю (рис. 3.10). Робітники мають доступ до налаштування IBAN, значення цього поля не відображається в додатку та шифрується під час збереження в базі даних.

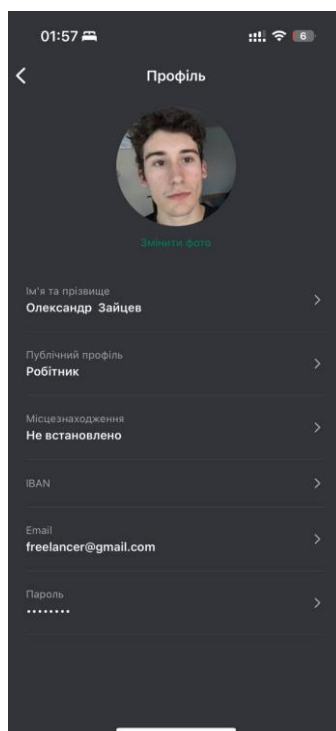


Рис. 3.10. Налаштування профілю

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

Робітники можуть налаштовувати місцезнаходження на мапі (рис. 3.11). Обрана локація буде слугувати позицією на мапі клієнтів під час пошуку робітників. Під час налаштування можливо переглядати власну локацію та автоматично направляти вибір на неї.

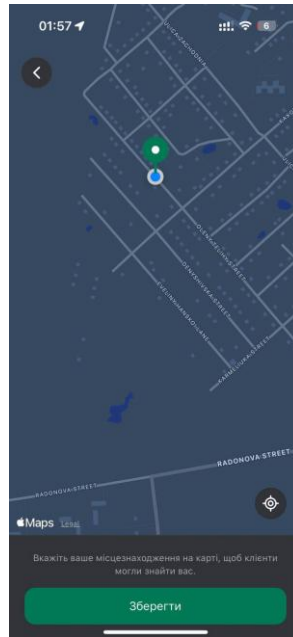


Рис. 3.11. Налаштування місцезнаходження

Для робітників доступна сторінка гаманця (рис. 3.12). Гаманець слугує накопиченням коштів, зароблених в результаті виконання робіт. Робітники можуть переглядати історію транзакцій у вигляді надходжень та переказів з відображенням дати транзакції та суми.

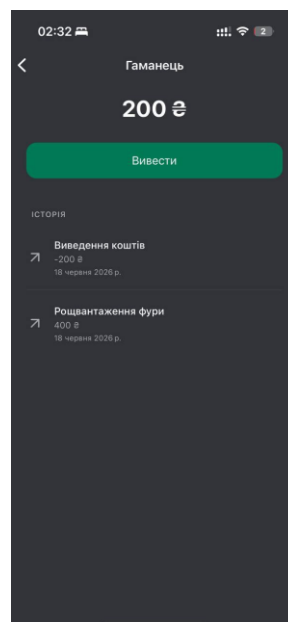


Рис. 3.12. Гаманець

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Робітники можуть переказувати кошти з платформи (рис. 3.13). Після введення суми та підтвердження з'явиться нова транзакція та баланс буде оновлений.

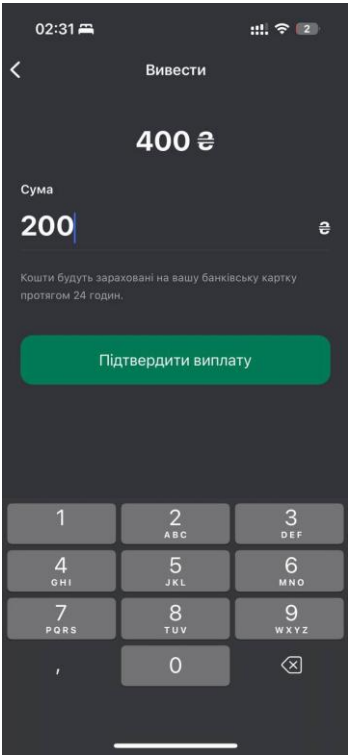


Рис. 3.13. Виведення коштів

Робітник має доступ до мапи доступних робіт, роботи відображаються з кольорами та картинками відповідно до їх категорій (рис. 3.14). Присутнє відображення локації користувача для кращої орієнтації. Роботи динамічно підвантажуються залежно від перегляду різних локацій на мапі. Роботи підвантажуються та кешуються фрагментами для оптимізації. Мапа для відображення вакансій доступна залежно від операційної системи на якій встановлений додаток. Для IOS використовується сервіс Apple Maps, для Android використовується сервіс Google Maps.

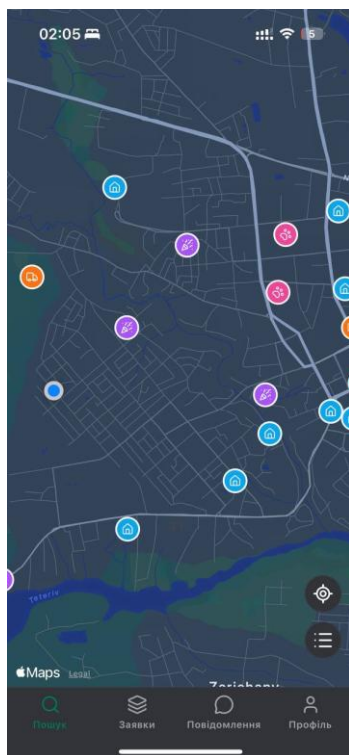


Рис. 3.14. Мапа робіт

При віддаленні, роботи збираються в кластери з кількістю робіт в певному фрагменті мапи (рис. 3.15). Роботи групуються, щоб не плутати користувача зовеликою кількістю інформації про роботи. Таким чином користувачі будуть переглядати роботи в конкретних частинах мапи.

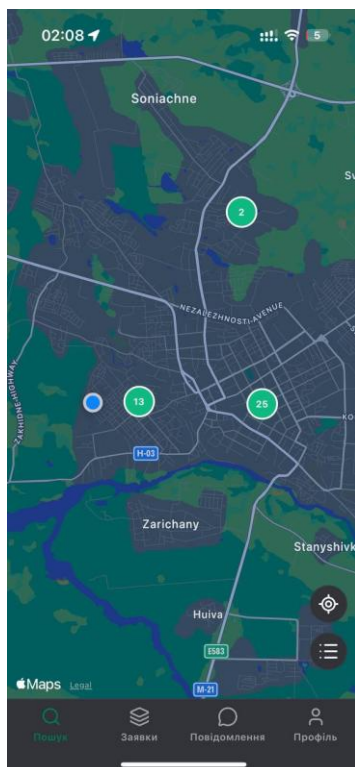


Рис. 3.15. Кластери робіт

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

При натисканні на певну роботу, мапа фокусується на обраній роботі та відображаються деталі з можливістю подати заявку (рис. 3.16). Відображаються уся додана клієнтом інформація про роботу, потрібна для ознайомлення.

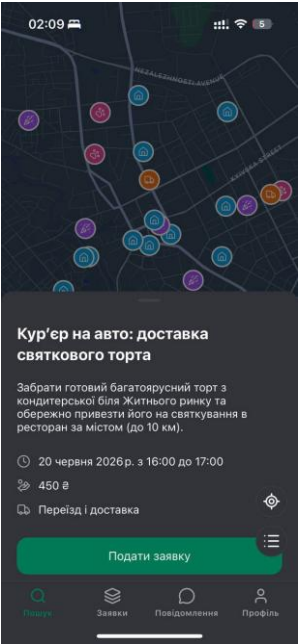


Рис. 3.16. Деталі роботи

Після натискання на кнопку подання заявки відображається форма заявки (рис. 3.17). Робітник вводить повідомлення та відправляє заявку до обраної роботи.

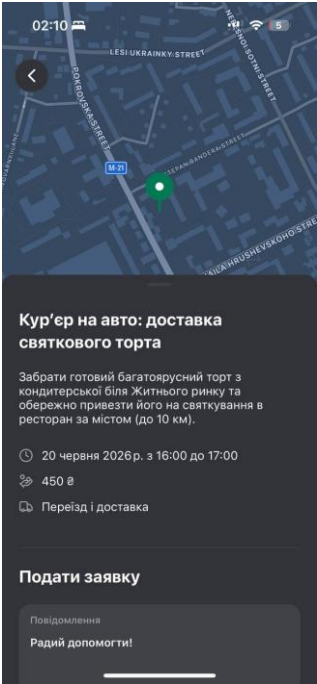


Рис. 3.17. Створення заявки

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

Створені заявки відображаються в списку на сторінці заявок (рис. 3.18).
Прийняті заявки відображаються за допомогою допоміжного знака про статус.

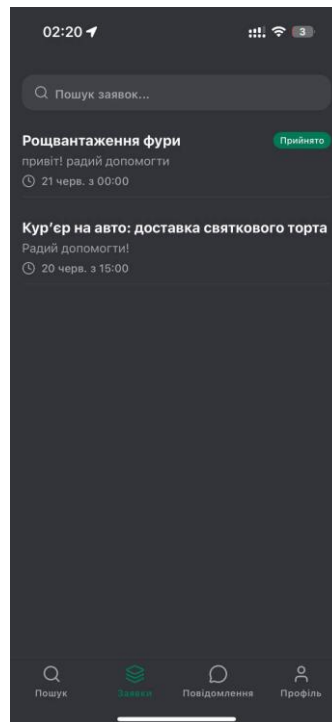


Рис. 3.18. Перегляд заявок робітника

При перегляді заявки робітник має доступ до всієї інформації про роботу а також її локацію (рис. 3.19). Також робітник може оновити або видалити заявку за бажанням.

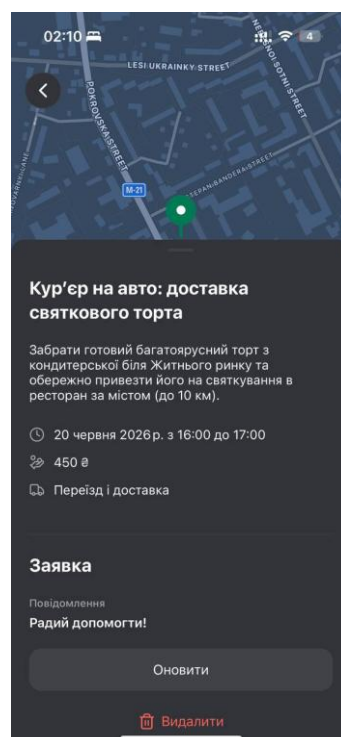


Рис. 3.19. Перегляд заявки робітника

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Також робітник може переглядати доступні роботи в режимі списку. Присутній доступ до пошуку за назвою, сортування та фільтрація за відстанню (рис. 3.20). Відстань розраховуються від локації користувача до локації роботи. Усі роботи за межами встановлено радіусу не відображаються.

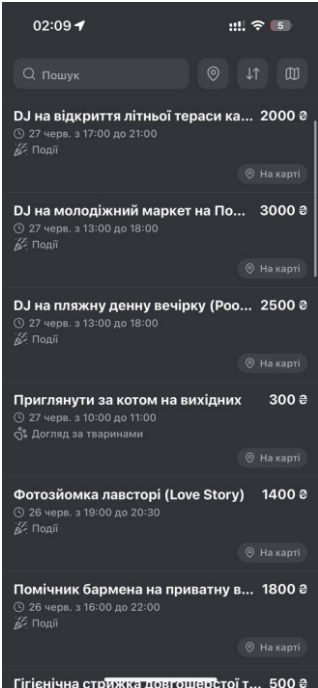


Рис. 3.20. Список доступних робіт

Клієнт має доступ до мапи робітників. Робітники відображаються в налаштованих локація у вигляді їх зображення (рис. 3.21).

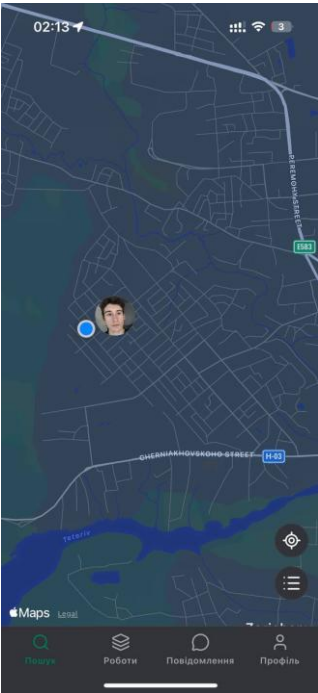


Рис. 3.21. Мапа з робітниками

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Також клієнти можуть переглядати робітників у вигляді списку (рис. 3.22). Присутній пошук, сортування, фільтрація за відстанню від клієнта до встановленої локації робітником

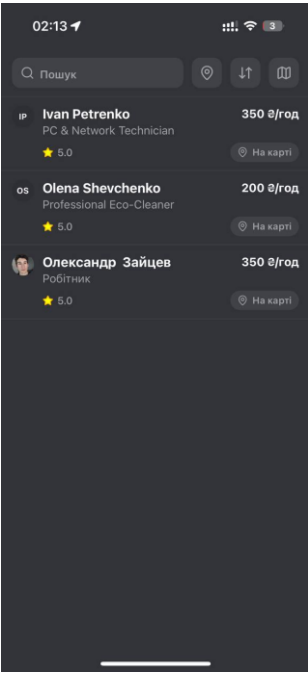


Рис. 3.22. Список з робітниками

Щоб створити роботу, клієнти відкривають сторінку з формою та обирають локацію та умови роботи (рис. 3.23). Клієнти можуть тимчасово згорнути форму для кращого відображення мапи та обрати потрібну локацію.

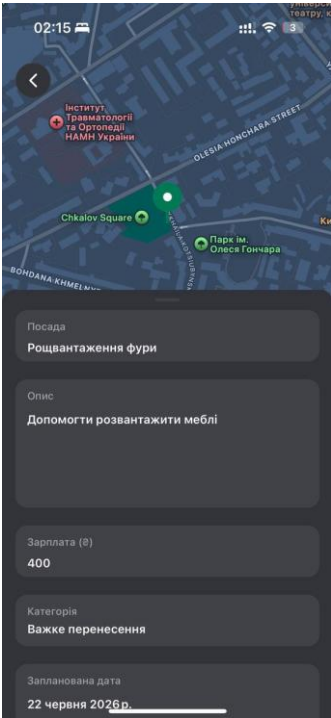


Рис. 3.23. Форма створення роботи

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

Створені роботи відображаються на сторінці робіт клієнта (рис. 3.24-3.25). Клієнти можуть переглядати сторінку роботи та оновлювати або видаляти обрану роботу.

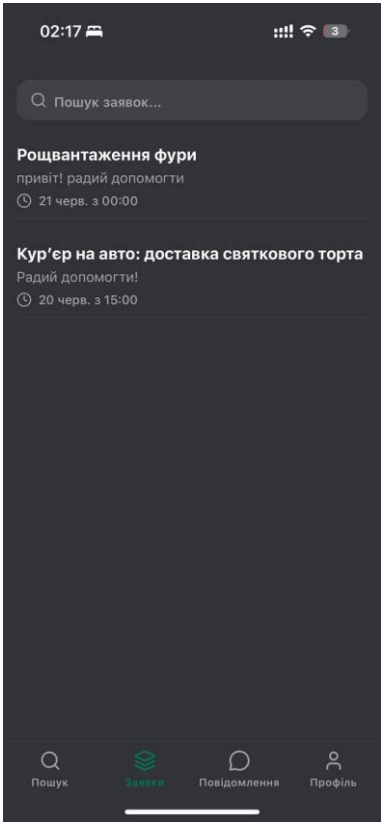


Рис. 3.24. Роботи клієнта

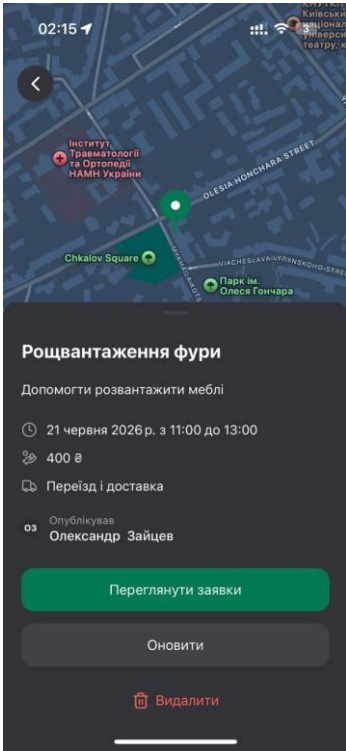


Рис. 3.25. Перегляд створеної роботи

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнт може переглядати заявки до власних робіт (рис. 3.26). На сторінці відображається список робітників, що подали заявку на обрану роботу. Також можна прочитати повідомлення додані робітниками для ознайомлення.

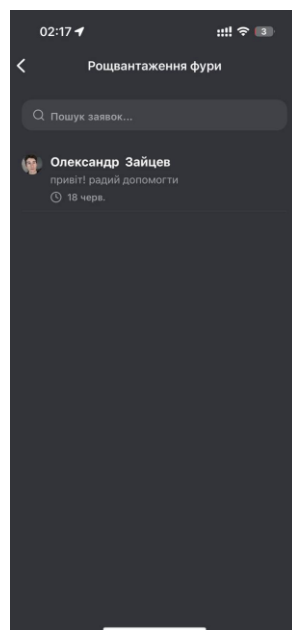


Рис. 3.26. Перегляд заявок

На сторінці заявки клієнт переглядає повідомлення та робітника (рис. 3.27). При потребі можна розпочати переписку з робітником (рис. 3.28). Додаток створить або відкриє існуючий чат між робітником та клієнтом та дозволить відправити повідомлення.

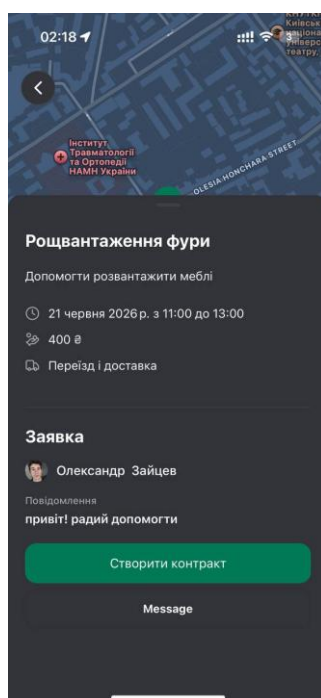


Рис. 3.27. Сторінка заявки

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

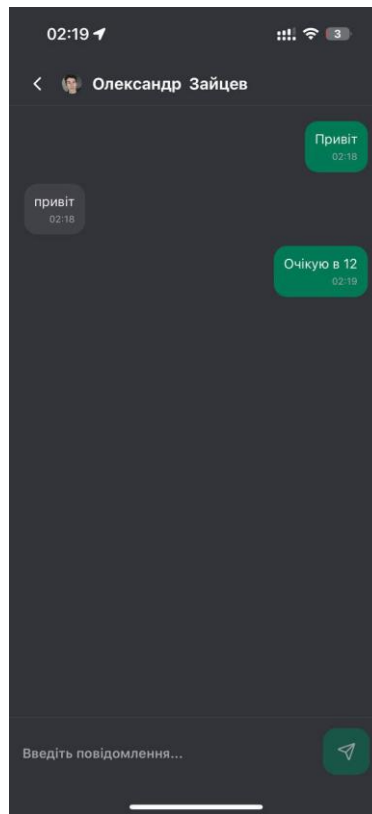


Рис. 3.28. Переписка з робітником

Клієнт може створити контракт з робітником. В контракті зазначаються умови виконання роботи (рис. 3.29). Контракт є документом що фіксує умови виконання роботи клієнта.

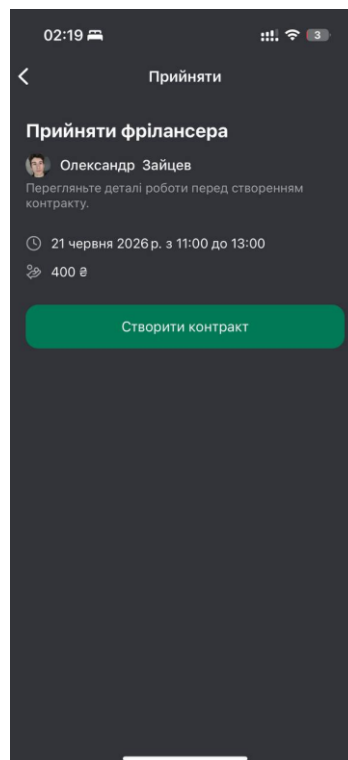


Рис. 3.29. Створення контракту

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Після створення, контракт отримує статус очікування.Поки контракт в цьому стані, клієнт може оновити умови контракту та очікувати підтвердження цих умов робітником (рис. 3.30).

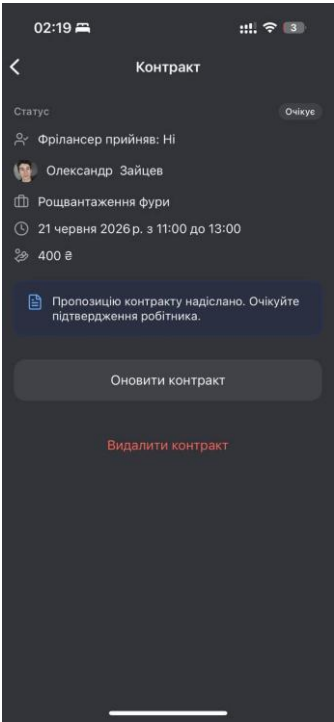


Рис. 3.30. Створений контракт

Робітник в свою чергу переглядає створений контракт та перевірити чи він згоден з умовами. Після перегляду робітник може підтвердити контракт для його подальшої оплати клієнтом (рис. 3.31).

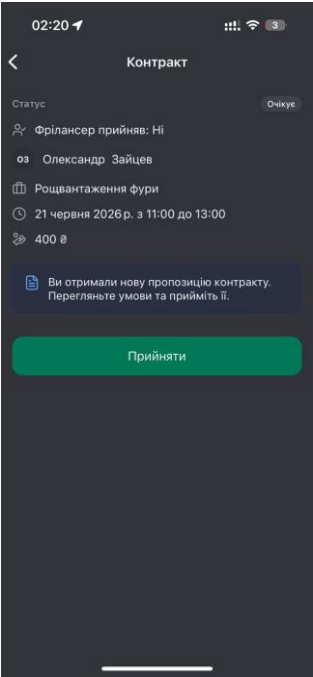


Рис. 3.31. Можливість робітника підтвердити контракт

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Після підтвердження контракту робітником клієнт бачить статус про підтвердження умов. Контракт може бути сплачений та активований для подальшого виконання (рис. 3.32).

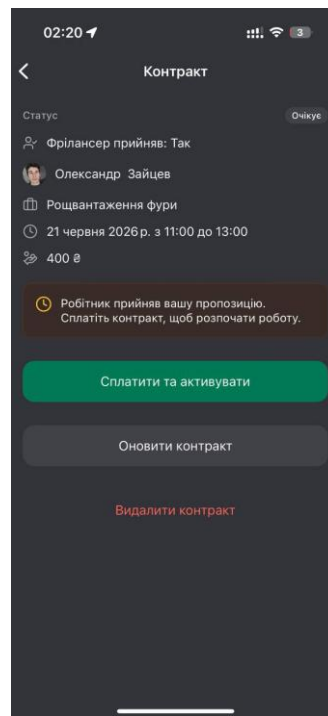


Рис. 3.32. Підтверджений контракт

Натиснувши на оплату контракту клієнт буде переадресований до оплати роботи в Monobank (рис. 3.33). Оплата відбувається безпечно на стороні Monobank, в результаті клієнт потрапляє на сторінку про успішну оплату контракту.

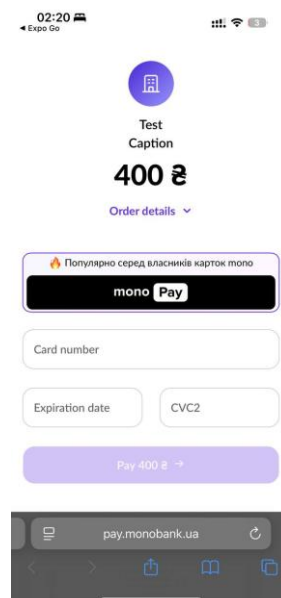


Рис. 3.33. Сплата контракту в Monobank

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішної оплати гроші будуть затримані в банку протягом 24 годин з можливістю скасувати оплату та контракт (рис. 3.34). Цей функціонал потрібен для можливості повернення коштів без втручання адміністрації платформи, оскільки кошти не знаходяться на рахунку платформи. Контракт не може бути скасований у випадку, коли робота є запланованою в межах найближчих 24 годин. Цей функціонал дозволяє уникнути можливості повернення коштів під час або після виконання роботи.

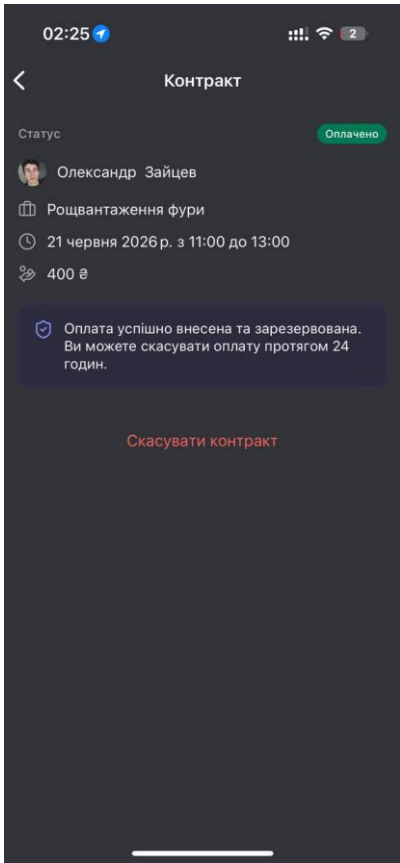


Рис. 3.34. Сплачений контракт

У наступні дві доби після оплати контракту гроші будуть нараховані на рахунок платформи та контракт буде активовано (рис. 3.35). Це статус свідчить про успішну угоду та можливість робітника приступити до виконання обов’язків зазначених у контракті.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

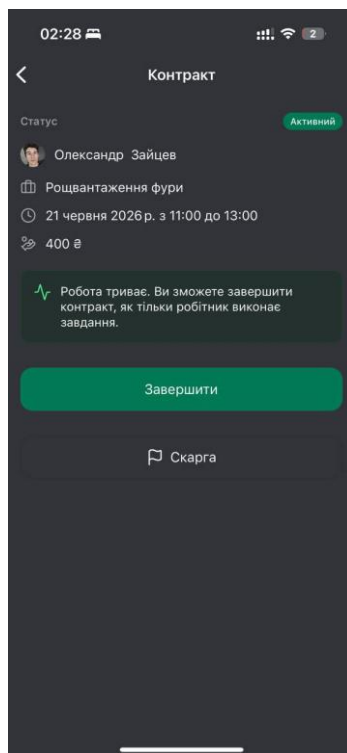


Рис. 3.35. Активований контракт

Після виконаної роботи клієнт завершує контракт та кошти надходять на віртуальний гаманець користувача (рис. 3.36). У випадку непорозумінь, клієнт та робітник можуть створити скаргу на контракт (рис. 3.37).

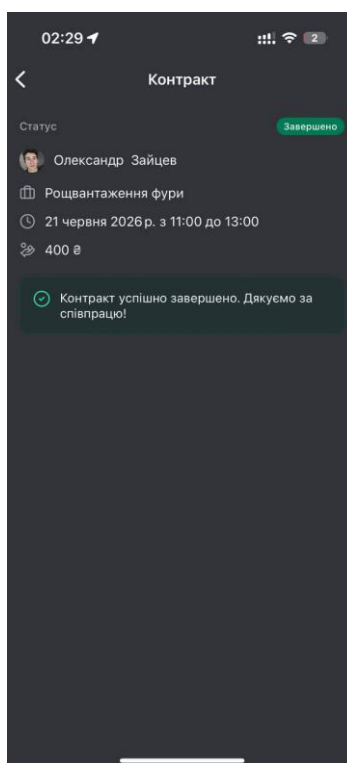


Рис. 3.36. Завершений контракт

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

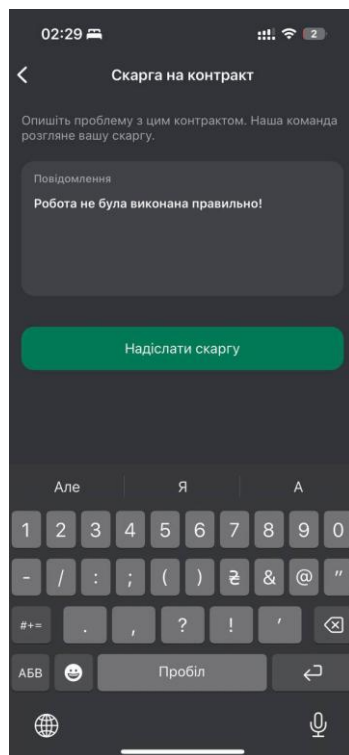


Рис. 3.37. Додавання скарги

Після виконаної роботи клієнт та робітник можуть залишити відгук один одному (рис. 3.38). Відгук містить в собі оцінку від одного до п'яти та додаткове повідомлення. Система відгуків дозволить користувачам отримувати користь від довготривалого використання платформи.

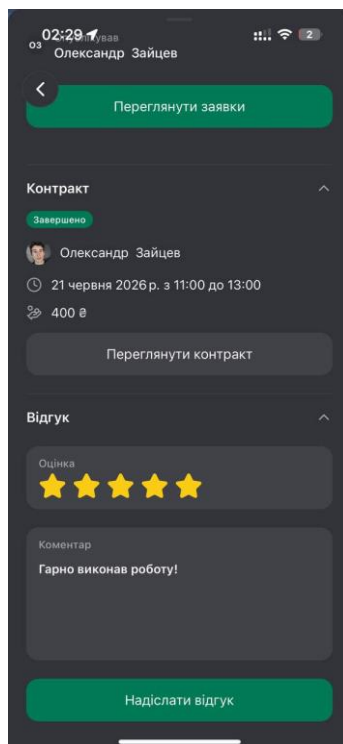


Рис. 3.38. Додавання відгуку

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

Виконані роботи та відгуки відображаються в профілі користувача (рис. 3.39-3.40). Клієнти можуть аналізувати історію робітника та його досвід з іншими користувачами. Це допоможе клієнтам бути впевненими у виборі правильно робітника.

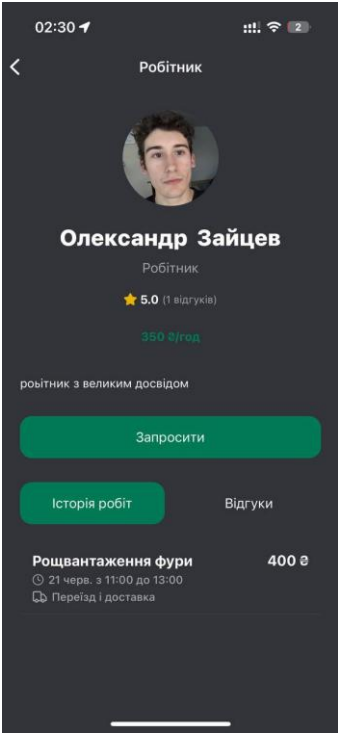


Рис. 3.39. Виконані роботи

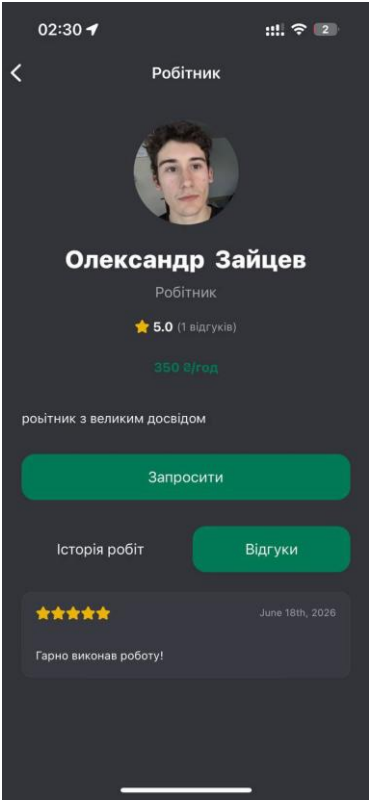


Рис. 3.40. Відгуки

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнти можуть запрошувати робітників виконувати роботи заповнюючи форму, повідомлення в чаті буде містити посилання на роботу (рис. 3.41-3.42). Робітників можна знайти на мапі або в списку та швидко запропонувати роботу. Так клієнти можуть не очікувати на заявки.

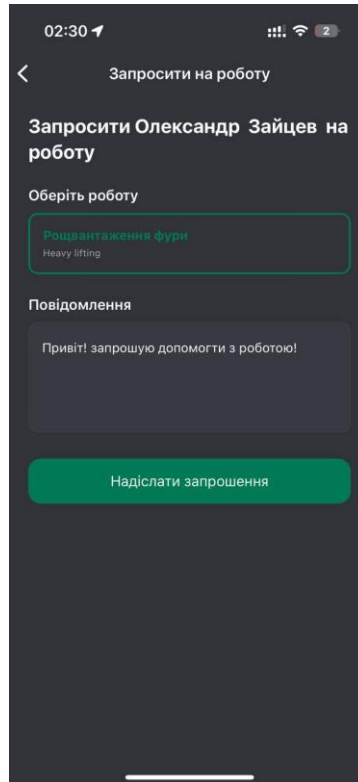


Рис. 3.41. Запрошення робітника

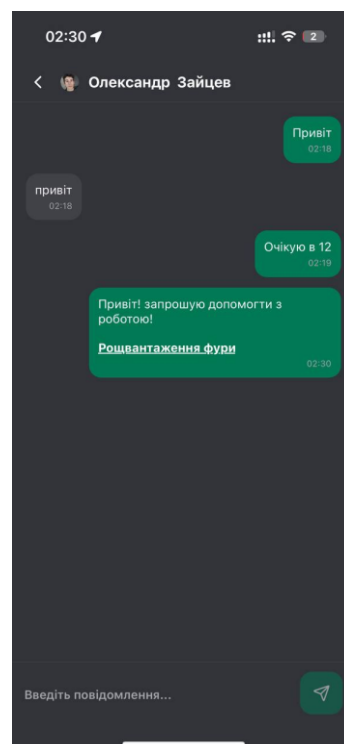


Рис. 3.42. Повідомлення з посиланням

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Структура інтерфейсу панелі адміністратора

На панелі адміністратора можна переглядати та керувати користувачами, роботами, контрактами (рис. 3.43-3.44). Перегляд списку кожної сутності підтримує пагінацію, пошук, сортування за більшістю полів. Це дозволяє максимально зручно знаходити необхідні дані для аналізу.

Ім'я ↑	Електронна пошта	Пошту підтверджено	Роль
Ivan Petrenko	ivan.dev@example.com	✓	Фрілансер
Natalia Melnyk	natalia.client@example.com	✓	Клієнт
Olena Shevchenko	olena.clean@example.com	✓	Фрілансер
Sergiy Kovalenko	sergiy.client@example.com	✓	Клієнт
Олександр Зайцев	client@gmail.com	✓	Клієнт
Олександр Зайцев	freelancer@gmail.com	✓	Фрілансер

Рис. 3.43. Користувачі системи

Назва ↑	Клієнт	Зарплата	Заплановано на	Статус
DJ на відкриття літньої тераси кав'ярня	Sergiy Kovalenko	2,000	27.06.2026 з 17:00 до 21:00	В процесі
DJ на молодіжний маркет на Подолі	Sergiy Kovalenko	3,000	27.06.2026 з 13:00 до 18:00	В процесі
DJ на пляжну денну вечірку (Pool Party)	Sergiy Kovalenko	2,500	27.06.2026 з 13:00 до 18:00	В процесі
Fix office router & setup LAN network	Sergiy Kovalenko	1,200	20.06.2026 з 9:00 до 15:00	В процесі
Full House Eco-Cleaning 2-bedroom flat	Natalia Melnyk	800	19.06.2026 з 8:00 до 13:00	Завершена
Активний вечірній вигул Лабрадора	Natalia Melnyk	300	21.06.2026 з 17:00 до 18:00	В процесі
Вигул спокійного Золотистого ретривера	Natalia Melnyk	250	21.06.2026 з 12:00 до 13:00	В процесі
Винесення будівництва після демонтажу плити	Natalia Melnyk	500	22.06.2026 з 17:00 до 18:30	В процесі
Вичісування ковтунів у довгошерстого кота	Natalia Melnyk	500	24.06.2026 з 14:00 до 16:00	В процесі

Рис. 3.44. Роботи

На сторінці контракту можна переглянути створений рахунок з Монобанк в межах угоди (рис. 3.45) [22]. Разонок підвантажується в реальному часі та відображає усі необхідні та актуальні дані про оплату.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

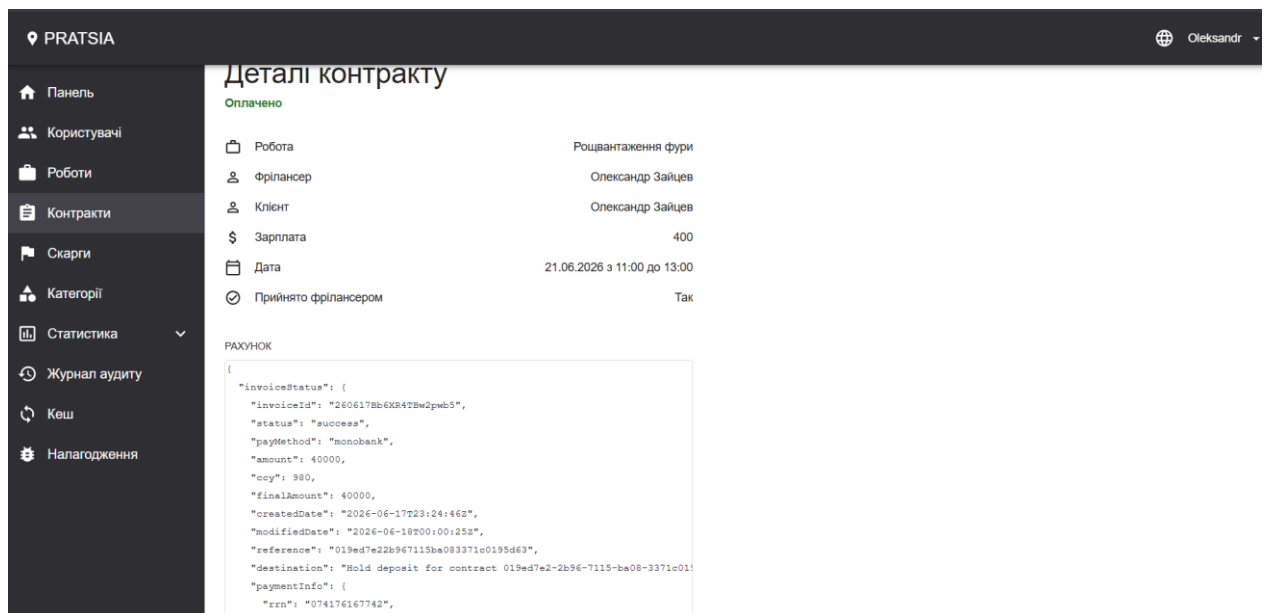


Рис. 3.45. Монобанк рахунок контракту

Адміністратор може переглядати скарги (рис. 3.46). Скарги містять посилання на контракт, користувача, що поскаржився, та повідомлення. Адміністратор може переглянути інформацію, зв'язатися з користувачами за допомогою електронної пошти та підтвердити або відхилити скаргу залежно від результату аналізу ситуації.

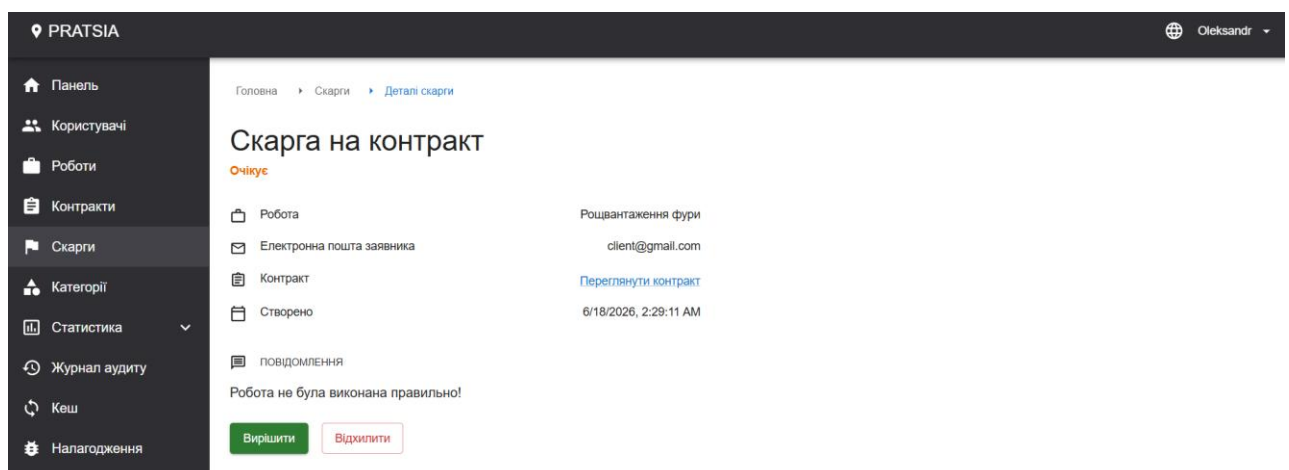


Рис. 3.46. Деталі скарги

Адміністратор має доступ до перегляду дій адміністраторів (рис. 3.47-3.48). Цей функціонал необхідний при роботі з вирішенням конфліктів між користувачами, таким чином можна відстежити усі операції та рішення адміністраторів. Функціонал покращує захист персональних даних користувачів, оскільки доступ до даних відслідковується.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

PRATSIA

Панель

Користувачі

Роботи

Контракти

Скарги

Категорії

Статистика

Журнал аудиту

Кеш

Налагодження

Головна

Журнал аудиту

Ім'я, пошта, дія

Ім'я	Електронна пошта	Часова мітка	Дії
Oleksandr	admin@gmail.com	6/18/2026, 4:12:11 AM	UserUpdated
Oleksandr	admin@gmail.com	6/18/2026, 2:26:40 AM	ContractPaymentFinalized

Rows per page: 101-2 of 2

Рис. 3.47. Дії адміністраторів

PRATSIA

Панель

Користувачі

Роботи

Контракти

Скарги

Категорії

Статистика

Журнал аудиту

Кеш

Налагодження

Головна

Журнал аудиту

Деталі запису аудиту

UserUpdated

Адміністратор

Oleksandr (admin@gmail.com)

019ed7ac-c873-74ab-a6f6-1de95af8b293

Часова мітка

6/18/2026, 4:12:11 AM

Повідомлення

Оновлено користувача з ID 11111111-1111-1111-1111-111111111111

"Role": 2,

"Email": "ivan.de@example.com"

"Role": 2,

"Email": "ivan.worker@example.com"

Рис. 3.48. Деталі дії адміністратора

Адмін панель відображає статистику платформи, таку як відношення заявок та робіт, відношення робітників та клієнтів, нові роботи (рис. 3.49).

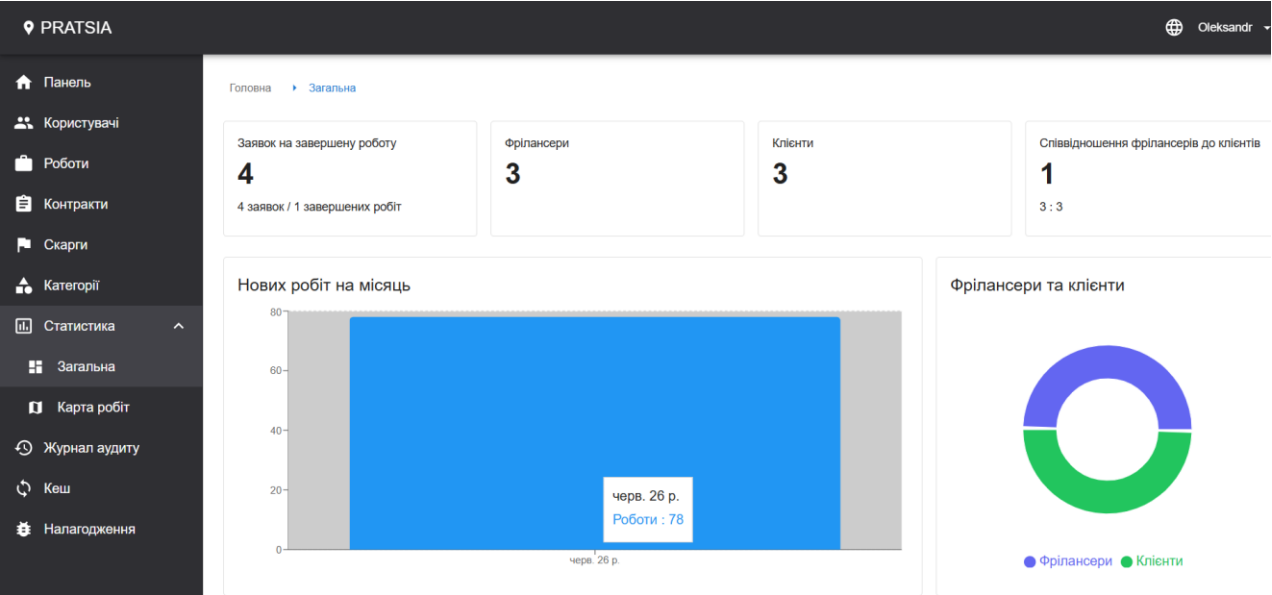


Рис. 3.49. Статистика системи

Також присутня статистика у вигляді мапи робіт, мапа дозволяє переглянути популярні категорії та середню зарплатню відносно геолокації (рис. 3.50-3.51). Завдяки алгоритму кластеризації, аналітика доступна для будь-яких масштабів: районів, міст та країн [23]. За допомогою цього функціоналу можна аналізувати ринок праці, виявляти тренди та відстежувати розповсюдження використання платформи.

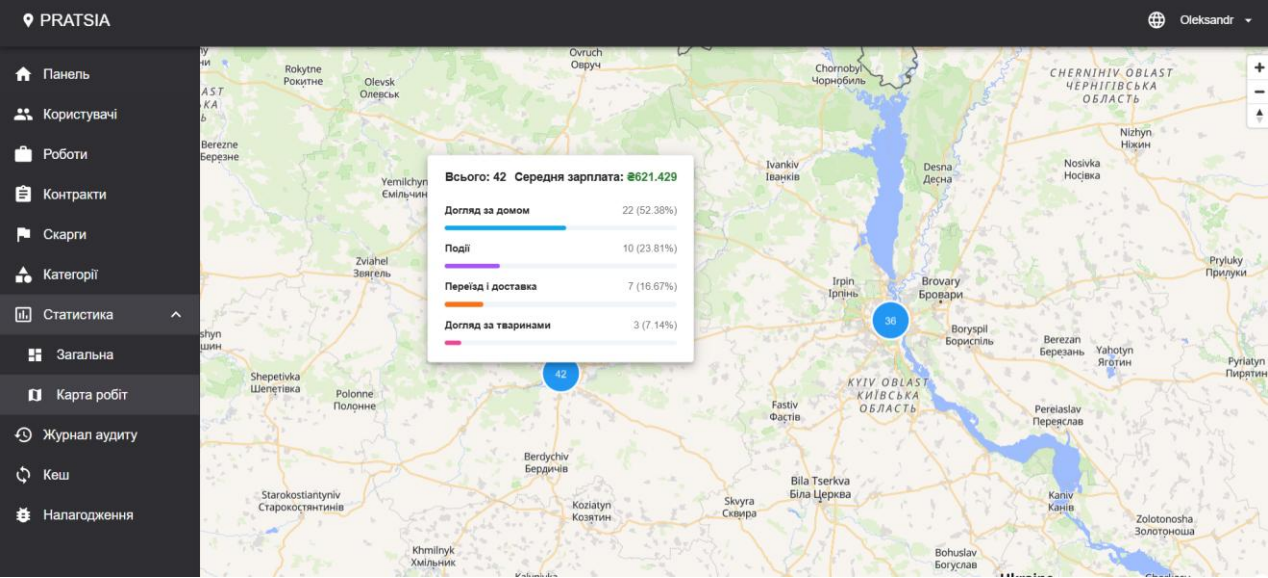


Рис. 3.50. Мапа статистики

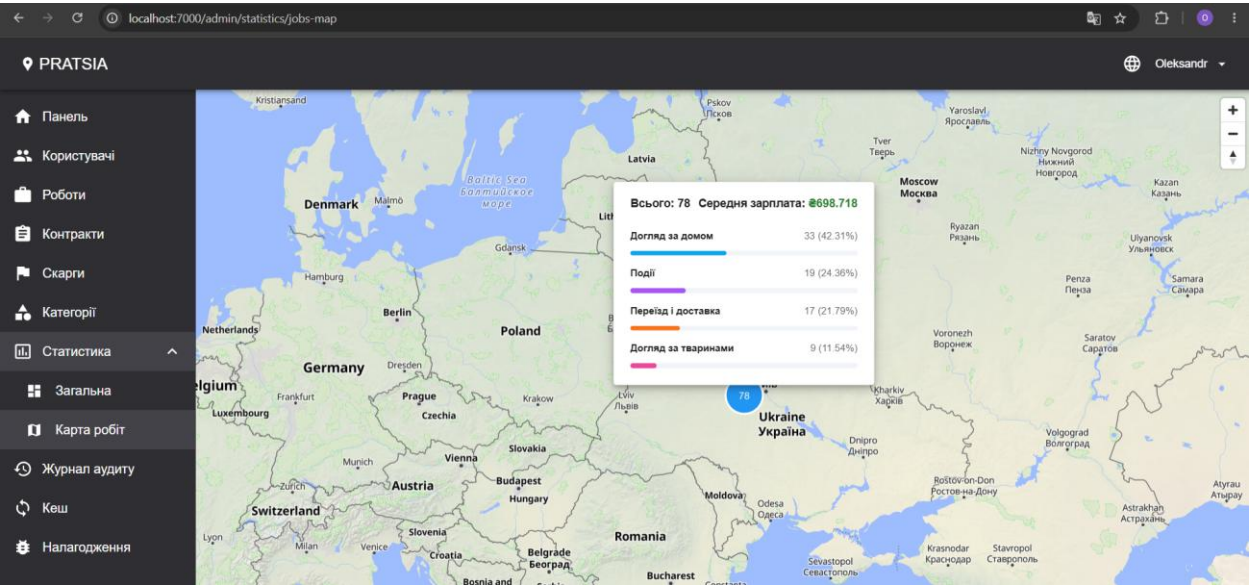


Рис. 3.51. Мапа статистики

3.4. Тестування вебплатформи

Відповідність реалізованого функціоналу технічним вимогам підтверджено шляхом автоматизованого тестування платформи засобами

.NET. Специфіка модульної монолітної архітектури дозволила організувати незалежне тестування кожного окремого модуля, суттєво спростивши пошук і локалізацію помилок. Юніт-тести прикладного рівня перевіряють обробники команд і запитів, які керують основними сценаріями взаємодії із системою [24].

Успішне виконання всього комплексу тестів свідчить про правильність реалізації ключових сценаріїв платформи, а також підтверджує відповідність розробленої системи архітектурним вимогам та обмеженням, визначеним на етапі проектування.

Процес та результати автоматизованого тестування системи відображено на рис. 3.52.

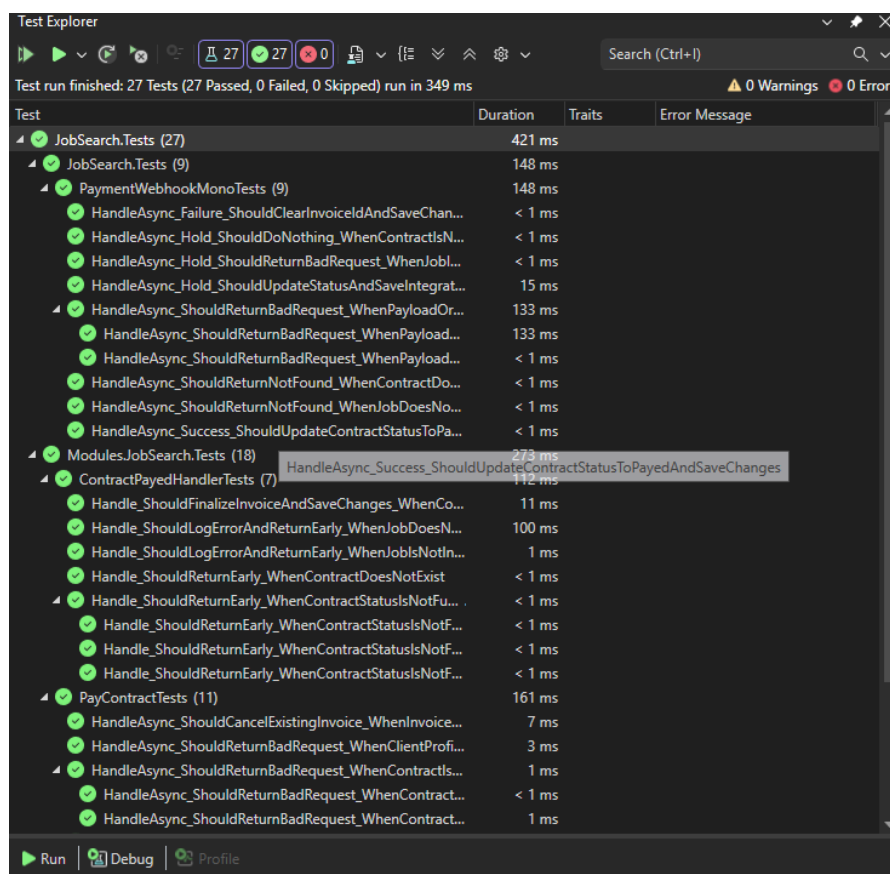


Рис. 3.52. Автоматизоване тестування

Окрім автоматизованого тестування серверної частини та API, було виконано комплексне ручне тестування мобільного додатка для операційних систем iOS та Android [25]. Головну увагу приділено коректності відображення графічного інтерфейсу на пристроях із різною діагоналлю

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

екрана, а також стабільності виконання базових функціональних сценаріїв. Окремим критично важливим етапом перевірки став аналіз взаємодії додатка з геолокаційними сервісами пристрою [26]. Зокрема, було протестовано логіку отримання дозволів від операційної системи про використання геолокації. Запит на використання геолокації додатком наведено на рис. 3.53. Підтвердження запиту наведено на рис. 3.54.

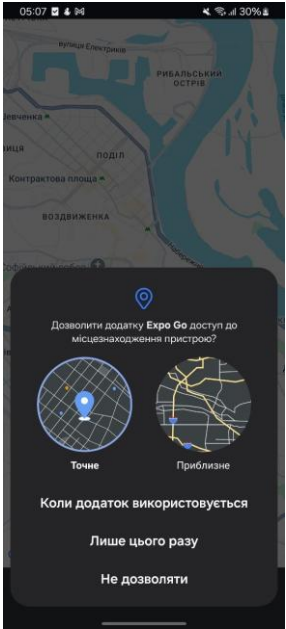


Рис. 3.53. Запит на використання геолокації

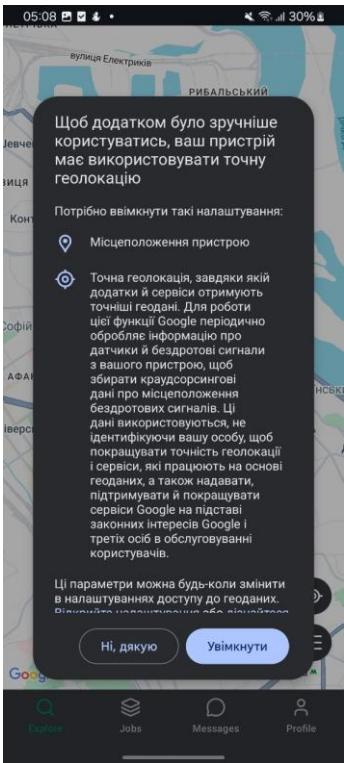


Рис. 3.54. Підтвердження запиту

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

Для визначення пропускної здатності розробленої системи було проведено стрес-тестування API. Основним інструментом для виконання цього етапу перевірки став інструмент Postman [27]. У ході тестування було налаштовано симуляцію екстремального навантаження шляхом генерації великої кількості одночасних запитів до ключових ендпоінтів API протягом фіксованого проміжку часу. Особливу увагу приділено аналізу поведінки серверної частини в умовах пікового трафіку. Головними критеріями, що відстежувалися під час експерименту, були час відповіді сервера на запити та динаміка зростання затримок. Було використано варіант тестування з поступовим збільшенням кількості запитів на секунду. Система продемонструвала високі результати з відсутністю помилок. Візуальний графік та результати тесту відображено на рис. 3.55.

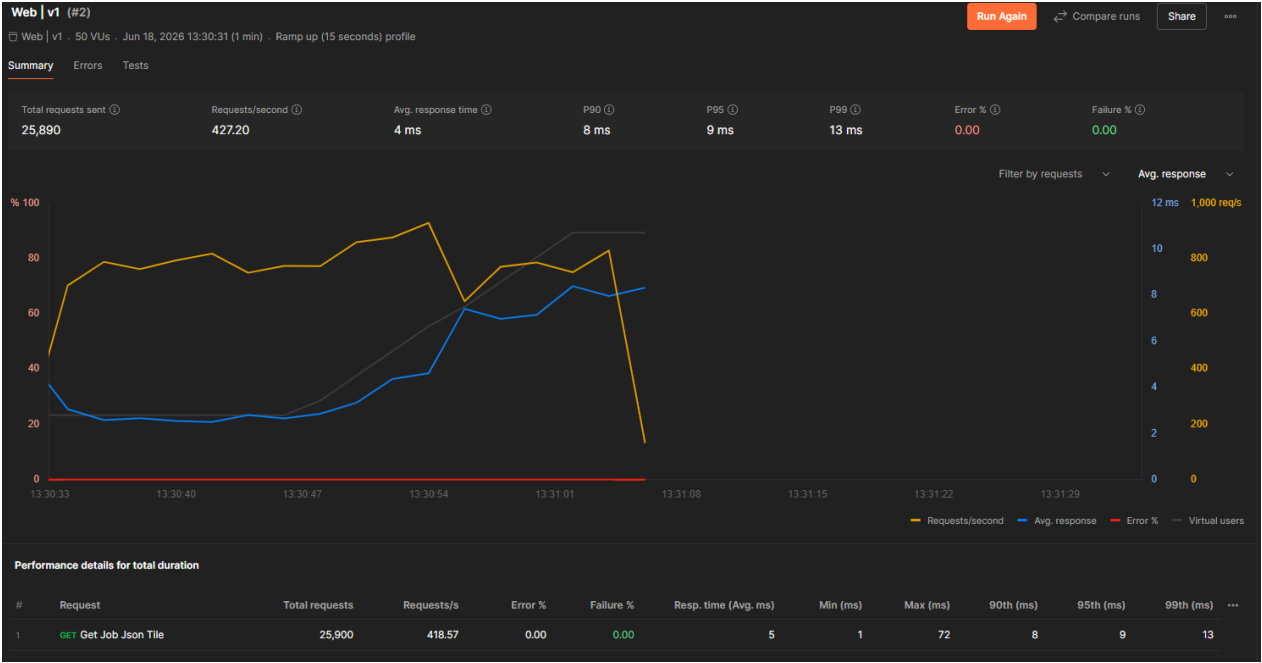


Рис. 3.55. тестування API системи

Завдяки серії тестування сервісу з’являється довіра до розробленого коду та гарантія стабільної роботи при великих навантаженнях. Після проведення необхідних перевірок сервіс може бути розгорнутий та використаний реальними користувачами. Ретельна перевірка спроможності програмного продукту значно вплине на репутацію платформи та лояльність користувачів.

Висновки до третього розділу

У третьому розділі викладено порядок розгортання, конфігурування та практичного застосування створеного сервісу, призначеного для пошуку та пропозиції короткочасної фізичної праці.

Проаналізовано процес розгортання інфраструктури з використанням технологій контейнеризації Docker та інструментів оркестрації на базі .NET Aspire, що дозволяє гнучко керувати залежностями серверних компонентів, розподіленого кешу Redis, сховища Azure Blob Storage та бази даних PostgreSQL з розширенням PostGIS. Налаштування конфігураційних файлів appsettings.json забезпечує ізоляцію середовищ розробки, тестування й експлуатації, створюючи стабільну й передбачувану поведінку системи при її оновленні чи первинному встановленні.

Висвітлено користувацький інтерфейс мобільного додатка React Native Expo та панелі для внутрішнього адміністрування платформи. Детально описано ключові сценарії використання мобільного додатку робітником та клієнтом. Продемонстровано можливості адміністрування платформи та перегляд аналітики.

Ефективність та надійність серверної частини підтверджено набором юніт-тестів. Натомість ручна перевірка мобільного клієнта на операційних системах iOS та Android дозволила детально проаналізувати коректність графічного інтерфейсу. Додатково виконане Postman стрес-тестування API підтвердило високу надійність та стабільність сервера під час високого трафіку.

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

ВИСНОВКИ

У результаті виконання бакалаврської випускної роботи було спроектовано та впроваджено сучасну геоінформаційну платформу, призначену для швидкої координації та безпечної взаємодії між замовниками й виконавцями одноразових фізичних робіт. На початковому етапі було проведено детальний аналіз ринку локальної зайнятості та досліджено наявні програмні продукти, що дозволило чітко виділити їхні архітектурні й функціональні особливості. Отримані висновки лягли в основу формування індивідуальних технічних вимог до власного цифрового рішення та довели доцільність його розробки.

В межах проєктування було обґрунтовано вибір архітектури та спроектовано оптимальну інформаційну структуру. Використання модульного підходу дозволило розділити логіку на рівні окремих схем даних та чітких контрактів взаємодії, забезпечивши високу швидкість розробки, простоту розгортання й надійний захист системи від заплутаних зв'язків коду.

Впроваджено серверну та клієнтську частини додатку. Створена система повністю покриває весь спектр можливостей, необхідних для координації замовників із шукачами роботи, та дозволяє ефективно задовільняти потреби обох сторін. Одночасно з цим було детально продемонстровано користувацький інтерфейс додатка та панель адміністрування, демонстрація показала зручність та інтуїтивність дизайну додатку.

Надійність сервісу перевірено шляхом комплексного тестування із залученням автоматизованих сценаріїв прикладного рівня, а також серії навмисних навантажень на систему та ручної перевірки.

Завдяки успішному виконанню цієї випускної кваліфікаційної роботи вдалося повністю досягти визначеної мети. Було розроблено сервіс інтерактивної мапи, який може слугувати інструментом розвитку кар'єр робітників та покращення побуту замовників послуг.

		<i>Зайцев О.І.</i>			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		<i>Морозов А.В.</i>				90
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pattern: Microservices Architecture. Microservices.io. URL: <https://microservices.io/patterns/microservices.html> (дата звернення: 18.06.2026).
2. Jovanovic M. What Is a Modular Monolith? Milan Jovanovic Blog. URL: <https://www.milanjovanovic.tech/blog/what-is-a-modular-monolith> (дата звернення: 16.06.2026).
3. Drescher A. Minimal APIs: Beyond the Basics. Medium. URL: <https://aubreydrescher.medium.com/minimal-apis-beyond-the-basics-99a075cb82d1> (дата звернення: 13.06.2026).
4. Jovanovic M. .NET Aspire: A Game Changer for Cloud-Native Development. Milan Jovanovic Blog. URL: <https://www.milanjovanovic.tech/blog/dotnet-aspire-a-game-changer-for-cloud-native-development> (дата звернення: 10.06.2026).
5. Sarwar F. S. React Native with Expo: A Developer's Tale. Medium. URL: <https://medium.com/@farrukh.ssarwar/react-native-with-expo-a-developers-tale-7afcaa499bdb> (дата звернення: 08.06.2026).
6. PostGIS Clustering with DBSCAN. Crunchy Data Blog. URL: <https://www.crunchydata.com/blog/postgis-clustering-with-dbscan> (дата звернення: 05.06.2026).
7. Azure Blob Storage in ASP.NET Core. .NET Intellect. 2024. 15 Sept. URL: <https://dotnetintellect.com/2024/09/15/azure-blob-storage-in-asp-net-core/> (дата звернення: 02.06.2026).
8. Distributed Caching in ASP.NET Core with Redis. Code with Mukesh. URL: <https://codewithmukesh.com/blog/distributed-caching-in-aspnet-core-with-redis/> (дата звернення: 31.05.2026).
9. Тестування API еквайрингу. Monobank API Docs. URL: <https://monobank.ua/api-docs/acquiring/dev/test/docs--testing> (дата звернення: 28.05.2026).

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

10. Single Rulebook Q&A: Public ID 2020_5477. European Banking Authority.
URL: https://www.eba.europa.eu/single-rule-book-qa/qna/view/publicId/2020_5477 (дата звернення: 26.05.2026).
11. PostgreSQL Documentation: 5.9. Schemas. PostgreSQL. URL: <https://www.postgresql.org/docs/current/ddl-schemas.html> (дата звернення: 23.05.2026).
12. Pattern: Transactional Outbox. Microservices.io. URL: <https://microservices.io/patterns/data/transactional-outbox.html> (дата звернення: 21.05.2026).
13. What is a Webhook? Red Hat. URL: <https://www.redhat.com/en/topics/automation/what-is-a-webhook> (дата звернення: 18.05.2026).
14. Mastering Maps in React Native: Custom Markers, Routes, Polygons, Geofencing. Medium: Simform Engineering. URL: <https://medium.com/simform-engineering/mastering-maps-in-react-native-custom-markers-routes-polygons-geofencing-f65a4bf74acc> (дата звернення: 15.05.2026).
15. Karki S. A Complete Guide to Quartz.NET Job Scheduling in ASP.NET Core. Dev.to. URL: <https://dev.to/satyakarki/a-complete-guide-to-quartznet-job-scheduling-in-aspnet-core-al0> (дата звернення: 13.05.2026).
16. What is Docker Compose? Docker Docs. URL: <https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-docker-compose/> (дата звернення: 10.05.2026).
17. Mahmud A. Server Side Rendering React with .NET Core. Medium. URL: <https://medium.com/@apakmahmud/server-side-rendering-react-with-net-core-1f3eaf8962fb> (дата звернення: 07.05.2026).
18. Appsettings in .NET Core: The Game Changer for Configurations. Medium: .NET FullStack Dev. URL: <https://dotnetfullstackdev.medium.com/appsettings-in-net-core-the-game-changer-for-configurations-a994d842e34c> (дата звернення: 05.05.2026).

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

19. What Is an Environment Variable? Kinsta Blog. URL: <https://kinsta.com/blog/what-is-an-environment-variable/> (дата звернення: 02.05.2026).
20. Using OpenAPI with .NET Core. Red Hat Developer. 2020. 16 Sept. URL: <https://developers.redhat.com/blog/2020/09/16/using-openapi-with-net-core> (дата звернення: 29.04.2026).
21. App Privacy Policy Guide. Usercentrics. URL: <https://usercentrics.com/guides/privacy-policy/app-privacy-policy/> (дата звернення: 27.04.2026).
22. Отримання статусу рахунку (GET /api/merchant/invoice/status). Monobank API Docs. URL: <https://monobank.ua/api-docs/acquiring/methods/ia/get--api--merchant--invoice--status> (дата звернення: 24.04.2026).
23. Density-Based Spatial Clustering Using PostGIS. Medium: Idea to Life. URL: <https://blog.ideatolife.me/density-based-spatial-clustering-using-postgis-%EF%B8%8F-3dd296cd2d3f> (дата звернення: 21.04.2026).
24. Unit Tests in .NET. Medium: ASP.NET. URL: <https://medium.com/asp-dotnet/unit-tests-in-net-f6f03313eaa5> (дата звернення: 19.04.2026).
25. Expo 2026: The Best Way to Build Cross-Platform Apps. MetaDesign Solutions Blog. URL: <https://metadesignsolutions.com/blog/expo-2026-the-best-way-to-build-cross-platform-apps> (дата звернення: 16.04.2026).
26. Expo SDK: Location. Expo Documentation. URL: <https://docs.expo.dev/versions/latest/sdk/location/> (дата звернення: 13.04.2026).
27. Postman API Performance Testing. Postman Blog. URL: <https://blog.postman.com/postman-api-performance-testing/> (дата звернення: 11.04.2026).
28. Iminza D. Sending Emails in .NET Using FluentEmail. Dev.to. URL: <https://dev.to/dianaiminza/sending-emails-in-net-using-fluentemail-mhc> (дата звернення: 08.04.2026).

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

29. Architecture Monolithique: Fondations, Forces et Limites. Architecti Blog. 2025. URL: <https://architecti.blog/en/articles/2025/011-architecture-monolithique---fondations-forces-et-limites/> (дата звернення: 05.04.2026).
30. What is gRPC? Postman Blog. URL: <https://blog.postman.com/what-is-grpc/> (дата звернення: 03.04.2026).

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

Лістинг програмного коду

Лістинг А.1:

```

public class PaymentWebhookMono : IEndpoint
{
    public void MapEndpoint(IEndpointRouteBuilder app)
    {
        app.MapPost("contracts/payment-webhook/mono", HandleAsync)
            .WithTags("Contracts")
            .WithName("PaymentWebhookMono");
    }
    private static async Task<Results<Ok, BadRequest<string>, NotFound<string>>> HandleAsync(
        MonoWebhookRequest payload,
        IContractRepository contractRepository,
        IJobRepository jobRepository,
        IJobSearchIntegrationEventService integrationEventService,
        IWalletService walletService,
        IUnitOfWork unitOfWork)
    {
        if (payload == null || string.IsNullOrEmpty(payload.InvoiceId))
        {
            return TypedResults.BadRequest("Invalid payload.");
        }

        var contract = await contractRepository.GetByInvoiceIdAsync(payload.InvoiceId);
        if (contract == null)
        {
            return TypedResults.NotFound($"Contract for invoice {payload.InvoiceId} not found.");
        }

        var job = await jobRepository.GetByIdAsync(contract.JobId);
        if (job is null)
        {
            return TypedResults.NotFound("Job not found");
        }

        switch (payload.Status.ToLowerInvariant())
        {
            case "hold":
                if (contract.Status != ContractStatus.Pending)
                    break;

                if (job.Status != JobStatus.Available)
                    return TypedResults.BadRequest("Only available jobs can have contracts activated.");

                contract.Status = ContractStatus.FundsHeld;
                job.ContractId = contract.Id;
                job.Status = JobStatus.InProgress;

                DateTime nextAvailableMidnight = DateTime.UtcNow.AddDays(2).Date;
                await integrationEventService.SaveEventAndChangesAsync(new ContractPayedEvent(contract.Id),
                    nextAvailableMidnight);
                break;

            case "failure":
                contract.PaymentInvoiceId = null;
                await unitOfWork.SaveChangesAsync();
                break;

            case "success":

```

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Морозов А.В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        contract.Status = ContractStatus.Payed;
        await unitOfWork.SaveChangesAsync();
        break;

    default:
        break;
}

return TypedResults.Ok();
}
}

public record ContractPaidEvent(Guid contractId) : IInternalEvent;

public record MonoWebhookRequest(
    string InvoiceId,
    string Status,
    int Amount,
    int Ссу,
    string ModifiedDate
);

```

Лістинг А.2:

```

public class PayContract : IEndpoint
{
    public void MapEndpoint(IEndpointRouteBuilder app)
    {
        app.MapPost("contracts/{id}/pay", HandleAsync)
            .WithTags("Contracts")
            .WithName("PayContract")
            .RequireClientAuth();
    }

    private static async Task<Results<Ok<MonoInvoiceResponse>, NotFound<string>, UnauthorizedHttpResult,
    BadRequest<string>>> HandleAsync(
        Guid id,
        IContractRepository contractRepository,
        IJobRepository jobRepository,
        IClientProfileRepository clientProfileRepository,
        IUnitOfWork unitOfWork,
        ClaimsPrincipal claimsPrincipal,
        IMonobankClient monobankClient)
    {
        var userId = claimsPrincipal.GetUserId();
        if (userId is null) return TypedResults.Unauthorized();

        var clientId = await clientProfileRepository.GetIdByUserIdAsync(userId.Value);
        if (clientId is null)
        {
            return TypedResults.BadRequest("Client profile not found");
        }

        var contract = await contractRepository.GetByIdAsync(id);
        if (contract is null)
        {
            return TypedResults.NotFound("Contract not found");
        }

        if (contract.ClientProfileId != clientId.Value)
        {
            return TypedResults.Unauthorized();
        }
    }
}

```

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

```

if (contract.Status != ContractStatus.Pending || !contract.IsAcceptedByFreelancer)
{
    return TypedResults.BadRequest("Only accepted pending contracts can be payed.");
}

var job = await jobRepository.GetByIdAsync(contract.JobId);
if (job is null)
{
    return TypedResults.NotFound("Job not found");
}

if (job.Status != JobStatus.Available)
{
    return TypedResults.BadRequest("Only available jobs can have contracts activated.");
}

if (!string.IsNullOrEmpty(contract.PaymentInvoiceId))
{
    await monobankClient.TryCancelHoldAsync(contract.PaymentInvoiceId);
    contract.PaymentInvoiceId = null;
}

var invoiceResponse = await monobankClient.CreateHoldInvoiceAsync(contract.Salary, contract.Id);
if (invoiceResponse is null)
{
    return TypedResults.BadRequest("Failed to create invoice.");
}

contract.PaymentInvoiceId = invoiceResponse.InvoiceId;
contract.PaymentDate = DateTime.UtcNow;

await unitOfWork.SaveChangesAsync();

return TypedResults.Ok(invoiceResponse);
}
}

```

Лістинг А.3:

```

public class ContractPayedHandler(
    IContractRepository contractRepository,
    IJobRepository jobRepository,
    IMonobankClient monobankClient,
    IUnitOfWork unitOfWork,
    ILogger<ContractPayedHandler> logger) : INotificationHandler<ContractPayedEvent>
{
    public async Task Handle(ContractPayedEvent notification, CancellationToken cancellationToken)
    {
        var contract = await contractRepository.GetByIdAsync(notification.contractId);
        if (contract is null)
        {
            return;
        }

        if (contract.Status != ContractStatus.FundsHeld || string.IsNullOrEmpty(contract.PaymentInvoiceId))
        {
            return;
        }

        var job = await jobRepository.GetByIdAsync(contract.JobId, cancellationToken);
        if (job is null || job.Status != JobStatus.InProgress)
        {
            logger.LogError("Job associated with contract {ContractId} is missing or no longer in progress. Payment is not captured.", contract.Id);
        }
    }
}

```

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

        return;
    }

    await monobankClient.FinalizeInvoiceAsync(contract.PaymentInvoiceId, contract.Salary);

    await unitOfWork.SaveChangesAsync(cancellationToken);
}
}

```

		Зайцев О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		99